```sql
/* Crie um banco de dados chamado Banco_do_RS */
create database IF NOT EXISTS  Banco_do_RS;

use Banco_do_RS;


/*
conta_corrente com o número, o saldo_atual, o nome do dono, número
do pix e o documento cpf/cnpj;
*/
create TABLE IF NOT EXISTS conta_corrente (
   cod_cc int not null AUTO_INCREMENT,
   numero_conta varchar(30),
   saldo_atual DECIMAL(10,2),
   nome VARCHAR(100),
   pix VARCHAR(50),
   documento VARCHAR(15),
   PRIMARY KEY (cod_cc)
);


/*
conta_poupança com o número, o saldo, o nome do dono, número do
pix e o documento cpf;
*/
create TABLE IF NOT EXISTS conta_poupanca (
   cod_cp int not null AUTO_INCREMENT,
   numero_conta varchar(30),
   saldo DECIMAL(10,2),
   pix VARCHAR(50),
   cpf char(11),
   cod_cc INTEGER,
   CONSTRAINT cod_cc FOREIGN KEY (cod_cc) REFERENCES
conta_corrente(cod_cc),
   PRIMARY KEY (cod_cp)
);


/*
```

```
cartao_credito com número, fatura_atual e número da conta do dono
do cartão de crédito;
*/
create TABLE IF NOT EXISTS conta_credito (
   cod_ccc int not null AUTO_INCREMENT,
   numero_conta varchar(30),
   fatura_atual DECIMAL(10,2),
   cod_fkcc INTEGER,
   CONSTRAINT cod_fkcc FOREIGN KEY (cod_fkcc) REFERENCES
conta_corrente(cod_cc),
   PRIMARY KEY (cod_ccc)
);


INSERT INTO conta_corrente (numero_conta,saldo_atual, nome, pix,
documento) VALUES ("0505050505", 10500.50, "Paulo jose",
"paulo@email.com", "01019283458");
INSERT INTO conta_corrente (numero_conta,saldo_atual, nome, pix,
documento) VALUES ("0505050580", 100.50, "maria",
"maria@email.com", "01019283458");
INSERT INTO conta_corrente (numero_conta,saldo_atual, nome, pix,
documento) VALUES ("092309213", 900.50, "lucio", "77566795040",
"77566795040");
INSERT INTO conta_corrente (numero_conta,saldo_atual, nome, pix,
documento) VALUES ("789238292", 5200.50, "fernadno",
"fernadno@email.com", "90236187007");
INSERT INTO conta_corrente (numero_conta,saldo_atual, nome, pix,
documento) VALUES ("989248912", 200.50, "mario", "29461306067m",
"29461306067");


INSERT INTO conta_poupanca (numero_conta,saldo, pix, cpf, cod_cc)
VALUES ("0505050505", 0.00,  "paulo@email.com", "01019283458", 1);
INSERT INTO conta_poupanca (numero_conta,saldo, pix, cpf, cod_cc)
VALUES ("0505050580", 0.00,  "maria@email.com", "01019283458", 2);
INSERT INTO conta_poupanca (numero_conta,saldo, pix, cpf, cod_cc)
VALUES ("092309213", 50.00,  "77566795040", "77566795040", 3);
INSERT INTO conta_poupanca (numero_conta,saldo, pix, cpf, cod_cc)
VALUES ("789238292", 0.00,  "fernadno@email.com", "90236187007",
4);
```

```sql
INSERT INTO conta_poupanca (numero_conta,saldo, pix, cpf, cod_cc)
VALUES ("989248912", 400.00,  "29461306067m", "29461306067", 5);


INSERT INTO conta_credito (numero_conta,fatura_atual, cod_fkcc)
VALUES ("0505050505", 100.50, 1);
INSERT INTO conta_credito (numero_conta,fatura_atual, cod_fkcc)
VALUES ("0505050580", 0.50, 2);
INSERT INTO conta_credito (numero_conta,fatura_atual, cod_fkcc)
VALUES ("092309213", 90.50, 3);
INSERT INTO conta_credito (numero_conta,fatura_atual, cod_fkcc)
VALUES ("789238292", 12200.50, 4);
INSERT INTO conta_credito (numero_conta,fatura_atual, cod_fkcc)
VALUES ("989248912", 400.50, 5);

/*
2) Agora crie um procedimento/função para saque onde retire o
valor do saldo da conta e atualize a tabela.
*/

CREATE FUNCTION `func_saque`(
   `valor_saque` DECIMAL(20,6),
   `saldo_conta` DECIMAL(20,6)
)
RETURNS FLOAT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
       DECLARE valor DECIMAL(20,2);
       DECLARE msg VARCHAR(100);
       if saldo_conta > valor_saque then
           SET valor = saldo_conta - valor_saque;
       ELSEIF saldo_conta < valor_saque then
           SET valor = 0;
           SET msg = "Valor de saque maior que saldo disponivel";
       END IF;

       RETURN valor;
```

```
END

CREATE PROCEDURE `proc_saque_conta_corrente`(
   IN `valor_saque` DECIMAL(20,6),
   IN `cod_cliente` INT
)
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
   DECLARE valor_calculado DECIMAL(10,2);

   SELECT round(func_saque(valor_saque,
conta_corrente.saldo_atual), 2) INTO @valor_calculado FROM
conta_corrente WHERE cod_cc = cod_cliente;

   if @valor_calculado != 0 then
       update conta_corrente SET conta_corrente.saldo_atual =
@valor_calculado where cod_cc = cod_cliente;
   else
       SELECT "Valor de saque maior que saldo atual";
   END IF;
END

call proc_saque_conta_corrente();


/* 3) Após isso, um procedimento/função de depósito. */
CREATE PROCEDURE `proc_deposito`(
   IN `valor_deposito` DECIMAL(20,6),
   IN `cod_cliente` INT
)
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
   DECLARE valor DECIMAL(10,2);
```

```
   SELECT conta_corrente.saldo_atual INTO @valor FROM
conta_corrente WHERE cod_cc = cod_cliente;

   update conta_corrente SET conta_corrente.saldo_atual = (@valor
+ valor_deposito) where cod_cc = cod_cliente;
END



/*
4) Após isso, um procedimento/função para transferência entre
contas. Lembre-se de utilizar IF ELSE nessa questão.
*/
CREATE PROCEDURE `proc_transferencia`(
   IN `de_conta` VARCHAR(150),
   IN `para_conta` VARCHAR(150),
   IN `valor` DECIMAL(20,6),
   IN `cod_cliente` INT
)
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN

   DECLARE _valor_cc DECIMAL(10, 2);
   DECLARE _valor_cp DECIMAL(10, 2);

   if de_conta = "corrente" && para_conta = "poupanca" then
       SELECT conta_corrente.saldo_atual INTO @_valor_cc FROM
conta_corrente WHERE cod_cc = cod_cliente;
       SELECT conta_poupanca.saldo INTO @_valor_cp FROM
conta_poupanca WHERE cod_cc = cod_cliente;

       update conta_corrente SET conta_corrente.saldo_atual =
(@_valor_cc - valor) where cod_cc = cod_cliente;
       update conta_poupanca SET conta_poupanca.saldo =
(@_valor_cp + valor) where cod_cp = cod_cliente;

   ELSEIF  de_conta = "poupanca" && para_conta = "corrente" then
```

```
       SELECT conta_corrente.saldo_atual INTO @_valor_cc FROM
conta_corrente WHERE cod_cc = cod_cliente;
       SELECT conta_poupanca.saldo INTO @_valor_cp FROM
conta_poupanca WHERE cod_cc = cod_cliente;

       update conta_corrente SET conta_corrente.saldo_atual =
(@_valor_cc + valor) where cod_cc = cod_cliente;
       update conta_poupanca SET conta_poupanca.saldo =
(@_valor_cp - valor) where cod_cp = cod_cliente;

   ELSE
       SELECT "Valores inconsistentes";
   END IF;
END

CALL proc_transferencia("corrente", "poupanca", 300.8, 1)


/*
5 a) Crie uma tabela chamada movimentação que contenha
o número da movimentação, o número da conta, o saldo
anterior e o saldo atual.
*/
create TABLE IF NOT EXISTS movimentacao (
   cod_m int not null AUTO_INCREMENT,
   numero_mov varchar(30),
   numero_conta varchar(30),
   saldo_anterior DECIMAL(10,2),
   saldo_atual DECIMAL(10,2),
   fkcod_ccm INTEGER,
   fkcod_cpm INTEGER,
   CONSTRAINT fkcod_ccm FOREIGN KEY (fkcod_ccm) REFERENCES
conta_corrente(cod_cc),
   CONSTRAINT fkcod_cpm FOREIGN KEY (fkcod_cpm) REFERENCES
conta_poupanca(cod_cp),
   PRIMARY KEY (cod_m)
);

INSERT INTO movimentacao (numero_mov, numero_conta,saldo_anterior,
saldo_atual, fkcod_ccm, fkcod_cpm) VALUES (005, "989248912", 200,
400.50, 1, 1);
```

```
INSERT INTO movimentacao (numero_mov, numero_conta,saldo_anterior,
saldo_atual, fkcod_ccm, fkcod_cpm) VALUES (006, "989248912", 300,
400.50, 2, 2);
INSERT INTO movimentacao (numero_mov, numero_conta,saldo_anterior,
saldo_atual, fkcod_ccm, fkcod_cpm) VALUES (007, "989248912", 400,
500.50, 3, 3);
INSERT INTO movimentacao (numero_mov, numero_conta,saldo_anterior,
saldo_atual, fkcod_ccm, fkcod_cpm) VALUES (008, "989248912", 230,
600.50, 4, 4);
INSERT INTO movimentacao (numero_mov, numero_conta,saldo_anterior,
saldo_atual, fkcod_ccm, fkcod_cpm) VALUES (009, "989248912", 220,
2400.50, 5, 5);


/*
5 b) Agora crie um trigger que após o update na
conta_corrente atualiza essa conta.
Dica: Você pode utilizar os dados da tabela conta
corrente no insert do trigger utilizando
OLD.numerodacoluna e o saldo atualizado como
NEW.saldo)
*/
CREATE TRIGGER `conta_corrente_before_update` BEFORE UPDATE ON
`conta_corrente` FOR EACH ROW BEGIN
   SET NEW.saldo_atual = OLD.saldo_atual;
END



/*
6) Crie uma função que liste contas com saldo maior que o valor
informado.
*/
CREATE PROCEDURE `proc_maior_saldo`(
   IN `valor` DECIMAL(20,6),
   IN `tipo_conta` VARCHAR(10)
)
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
```

```
COMMENT ''
BEGIN
	case
		when tipo_conta = "cc" then
			SELECT * FROM conta_corrente WHERE
conta_corrente.saldo_atual > valor;
		when tipo_conta = "cp" then
			SELECT * FROM conta_poupanca WHERE
conta_poupanca.saldo > valor;
		when tipo_conta = "cred" then
			SELECT * FROM conta_credito WHERE
conta_credito.fatura_atual > valor;
		ELSE SELECT 'Verifique valores de entrada';
	END case;
END

CALL proc_maior_saldo(10000, 'cp');


/*
7) Crie um procedimento que modifica o saldo de todas as
contas com o valor informado; Dica: Remova o safe updates
nas configs.
*/
CREATE PROCEDURE `proc_altera_saldo`(
   IN `valor` DECIMAL(20,6),
   IN `conta` VARCHAR(50)
)
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
   UPDATE conta_corrente SET conta_corrente.saldo_atual = valor
WHERE conta_corrente.numero_conta = conta;
   UPDATE conta_poupanca SET conta_poupanca.saldo = valor WHERE
conta_poupanca.numero_conta = conta;
   UPDATE conta_credito SET conta_credito.fatura_atual = valor;
END
```

```
/*
8) Agora crie os seguintes triggers:
a) Ao deletar uma conta apague também as
movimentações salvas (AFTER)
*/
CREATE TRIGGER `conta_corrente_after_delete` AFTER DELETE ON
`conta_corrente` FOR EACH ROW BEGIN
   DELETE FROM movimentacao
   WHERE movimentacao.fkcod_ccm = OLD.cod_cc;
END

DELETE FROM conta_corrente WHERE conta_corrente.cod_cc = 1;


/*
8 b) Ao criar uma conta salvar um registro na tabela
movimentação com saldo antigo e atual igual a 0
(AFTER)
*/
CREATE TRIGGER `conta_corrente_after_insert` AFTER INSERT ON
`conta_corrente` FOR EACH ROW BEGIN

   INSERT INTO movimentacao (numero_mov,
numero_conta,saldo_anterior, saldo_atual, fkcod_ccm, fkcod_cpm)
   VALUES (round(RAND()*100,2), NEW.numero_conta,  0.00,
NEW.saldo_atual, NEW.cod_cc, 5);

END

INSERT INTO conta_corrente (numero_conta,saldo_atual, nome, pix,
documento) VALUES ("9892492312", 850, "kaio", "00461306067",
"00461306067");


/*
8 c) Antes de cadastrar uma nova conta verifique se o saldo
informado é igual a 0, se não: atualize para 0 (BEFORE,
utilize IF ELSE)
*/
CREATE TRIGGER `check_insert_conta_corrente` BEFORE INSERT ON
`conta_corrente` FOR EACH ROW BEGIN
```

```
   if NEW.saldo_atual = 0 then
       INSERT INTO conta_corrente (numero_conta,saldo_atual, nome,
pix, documento)
           VALUES (NEW.numero_conta, NEW.saldo_atual, NEW.nome,
NEW.pix, NEW.documento);
   ELSE
       INSERT INTO conta_corrente (numero_conta,saldo_atual, nome,
pix, documento)
           VALUES (NEW.numero_conta, 0.00, NEW.nome, NEW.pix,
NEW.documento);
   END if;

END


/*
8 d) Antes de deletar uma conta, salvar um registro em
movimentação de que a conta está zerada (BEFORE)
*/
CREATE TRIGGER `check_valor_movimentacao` BEFORE DELETE ON
`conta_corrente` FOR EACH ROW BEGIN
   INSERT INTO movimentacao (numero_mov,
numero_conta,saldo_anterior, saldo_atual, fkcod_ccm, fkcod_cpm)
       VALUES (round(RAND()*100,2), old.numero_conta,
OLD.saldo_atual, 0.00, old.cod_cc, 5);
END


/* Crie um banco de dados chamado Banco_do_RS */
create database IF NOT EXISTS  Banco_do_RS;

use Banco_do_RS;


/*
conta_corrente com o número, o saldo_atual, o nome do dono, número
do pix e o documento cpf/cnpj;
*/
create TABLE IF NOT EXISTS conta_corrente (
    cod_cc int not null AUTO_INCREMENT,
```

```
    numero_conta varchar(30),
    saldo_atual DECIMAL(10,2),
    nome VARCHAR(100),
    pix VARCHAR(50),
    documento VARCHAR(15),
    PRIMARY KEY (cod_cc)
);


/*
conta_poupança com o número, o saldo, o nome do dono, número do
pix e o documento cpf;
 */
create TABLE IF NOT EXISTS conta_poupanca (
    cod_cp int not null AUTO_INCREMENT,
    numero_conta varchar(30),
    saldo DECIMAL(10,2),
    pix VARCHAR(50),
    cpf char(11),
    cod_cc INTEGER,
    CONSTRAINT cod_cc FOREIGN KEY (cod_cc) REFERENCES
conta_corrente(cod_cc),
    PRIMARY KEY (cod_cp)
);


/*
cartao_credito com número, fatura_atual e número da conta do dono
do cartão de crédito;
*/
create TABLE IF NOT EXISTS conta_credito (
    cod_ccc int not null AUTO_INCREMENT,
    numero_conta varchar(30),
    fatura_atual DECIMAL(10,2),
    cod_fkcc INTEGER,
    CONSTRAINT cod_fkcc FOREIGN KEY (cod_fkcc) REFERENCES
conta_corrente(cod_cc),
    PRIMARY KEY (cod_ccc)
);
```

```
INSERT INTO conta_corrente (numero_conta,saldo_atual, nome, pix,
documento) VALUES ("0505050505", 10500.50, "Paulo jose",
"paulo@email.com", "01019283458");
INSERT INTO conta_corrente (numero_conta,saldo_atual, nome, pix,
documento) VALUES ("0505050580", 100.50, "maria",
"maria@email.com", "01019283458");
INSERT INTO conta_corrente (numero_conta,saldo_atual, nome, pix,
documento) VALUES ("092309213", 900.50, "lucio", "77566795040",
"77566795040");
INSERT INTO conta_corrente (numero_conta,saldo_atual, nome, pix,
documento) VALUES ("789238292", 5200.50, "fernadno",
"fernadno@email.com", "90236187007");
INSERT INTO conta_corrente (numero_conta,saldo_atual, nome, pix,
documento) VALUES ("989248912", 200.50, "mario", "29461306067m",
"29461306067");


INSERT INTO conta_poupanca (numero_conta,saldo, pix, cpf, cod_cc)
VALUES ("0505050505", 0.00,  "paulo@email.com", "01019283458", 1);
INSERT INTO conta_poupanca (numero_conta,saldo, pix, cpf, cod_cc)
VALUES ("0505050580", 0.00,  "maria@email.com", "01019283458", 2);
INSERT INTO conta_poupanca (numero_conta,saldo, pix, cpf, cod_cc)
VALUES ("092309213", 50.00,  "77566795040", "77566795040", 3);
INSERT INTO conta_poupanca (numero_conta,saldo, pix, cpf, cod_cc)
VALUES ("789238292", 0.00,  "fernadno@email.com", "90236187007",
4);
INSERT INTO conta_poupanca (numero_conta,saldo, pix, cpf, cod_cc)
VALUES ("989248912", 400.00,  "29461306067m", "29461306067", 5);


INSERT INTO conta_credito (numero_conta,fatura_atual, cod_fkcc)
VALUES ("0505050505", 100.50, 1);
INSERT INTO conta_credito (numero_conta,fatura_atual, cod_fkcc)
VALUES ("0505050580", 0.50, 2);
INSERT INTO conta_credito (numero_conta,fatura_atual, cod_fkcc)
VALUES ("092309213", 90.50, 3);
INSERT INTO conta_credito (numero_conta,fatura_atual, cod_fkcc)
VALUES ("789238292", 12200.50, 4);
INSERT INTO conta_credito (numero_conta,fatura_atual, cod_fkcc)
VALUES ("989248912", 400.50, 5);
```

```
/*
2) Agora crie um procedimento/função para saque onde retire o
valor do saldo da conta e atualize a tabela.
*/

CREATE FUNCTION `func_saque`(
     `valor_saque` DECIMAL(20,6),
     `saldo_conta` DECIMAL(20,6)
)
RETURNS FLOAT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
          DECLARE valor DECIMAL(20,2);
          DECLARE msg VARCHAR(100);
          if saldo_conta > valor_saque then
               SET valor = saldo_conta - valor_saque;
          ELSEIF saldo_conta < valor_saque then
               SET valor = 0;
               SET msg = "Valor de saque maior que saldo
disponivel";
          END IF;

          RETURN valor;
END

CREATE PROCEDURE `proc_saque_conta_corrente`(
     IN `valor_saque` DECIMAL(20,6),
     IN `cod_cliente` INT
)
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
     DECLARE valor_calculado DECIMAL(10,2);
```

```
     SELECT round(func_saque(valor_saque,
conta_corrente.saldo_atual), 2) INTO @valor_calculado FROM
conta_corrente WHERE cod_cc = cod_cliente;

     if @valor_calculado != 0 then
          update conta_corrente SET conta_corrente.saldo_atual =
@valor_calculado where cod_cc = cod_cliente;
     else
          SELECT "Valor de saque maior que saldo atual";
     END IF;
END

call proc_saque_conta_corrente();



/* 3) Após isso, um procedimento/função de depósito. */
CREATE PROCEDURE `proc_deposito`(
     IN `valor_deposito` DECIMAL(20,6),
     IN `cod_cliente` INT
)
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
     DECLARE valor DECIMAL(10,2);

     SELECT conta_corrente.saldo_atual INTO @valor FROM
conta_corrente WHERE cod_cc = cod_cliente;

     update conta_corrente SET conta_corrente.saldo_atual =
(@valor + valor_deposito) where cod_cc = cod_cliente;
END



/*
4) Após isso, um procedimento/função para transferência entre
contas. Lembre-se de utilizar IF ELSE nessa questão.
*/
CREATE PROCEDURE `proc_transferencia`(
```

```
     IN `de_conta` VARCHAR(150),
     IN `para_conta` VARCHAR(150),
     IN `valor` DECIMAL(20,6),
     IN `cod_cliente` INT
)
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN

     DECLARE _valor_cc DECIMAL(10, 2);
     DECLARE _valor_cp DECIMAL(10, 2);

     if de_conta = "corrente" && para_conta = "poupanca" then
          SELECT conta_corrente.saldo_atual INTO @_valor_cc FROM
conta_corrente WHERE cod_cc = cod_cliente;
          SELECT conta_poupanca.saldo INTO @_valor_cp FROM
conta_poupanca WHERE cod_cc = cod_cliente;

          update conta_corrente SET conta_corrente.saldo_atual =
(@_valor_cc - valor) where cod_cc = cod_cliente;
          update conta_poupanca SET conta_poupanca.saldo =
(@_valor_cp + valor) where cod_cp = cod_cliente;

     ELSEIF  de_conta = "poupanca" && para_conta = "corrente" then
          SELECT conta_corrente.saldo_atual INTO @_valor_cc FROM
conta_corrente WHERE cod_cc = cod_cliente;
          SELECT conta_poupanca.saldo INTO @_valor_cp FROM
conta_poupanca WHERE cod_cc = cod_cliente;

          update conta_corrente SET conta_corrente.saldo_atual =
(@_valor_cc + valor) where cod_cc = cod_cliente;
          update conta_poupanca SET conta_poupanca.saldo =
(@_valor_cp - valor) where cod_cp = cod_cliente;

     ELSE
          SELECT "Valores inconsistentes";
     END IF;
END
```

```
CALL proc_transferencia("corrente", "poupanca", 300.8, 1)


/*
5 a) Crie uma tabela chamada movimentação que contenha
o número da movimentação, o número da conta, o saldo
anterior e o saldo atual.
*/
create TABLE IF NOT EXISTS movimentacao (
    cod_m int not null AUTO_INCREMENT,
    numero_mov varchar(30),
    numero_conta varchar(30),
    saldo_anterior DECIMAL(10,2),
    saldo_atual DECIMAL(10,2),
    fkcod_ccm INTEGER,
    fkcod_cpm INTEGER,
    CONSTRAINT fkcod_ccm FOREIGN KEY (fkcod_ccm) REFERENCES
conta_corrente(cod_cc),
    CONSTRAINT fkcod_cpm FOREIGN KEY (fkcod_cpm) REFERENCES
conta_poupanca(cod_cp),
    PRIMARY KEY (cod_m)
);

INSERT INTO movimentacao (numero_mov, numero_conta,saldo_anterior,
saldo_atual, fkcod_ccm, fkcod_cpm) VALUES (005, "989248912", 200,
400.50, 1, 1);
INSERT INTO movimentacao (numero_mov, numero_conta,saldo_anterior,
saldo_atual, fkcod_ccm, fkcod_cpm) VALUES (006, "989248912", 300,
400.50, 2, 2);
INSERT INTO movimentacao (numero_mov, numero_conta,saldo_anterior,
saldo_atual, fkcod_ccm, fkcod_cpm) VALUES (007, "989248912", 400,
500.50, 3, 3);
INSERT INTO movimentacao (numero_mov, numero_conta,saldo_anterior,
saldo_atual, fkcod_ccm, fkcod_cpm) VALUES (008, "989248912", 230,
600.50, 4, 4);
INSERT INTO movimentacao (numero_mov, numero_conta,saldo_anterior,
saldo_atual, fkcod_ccm, fkcod_cpm) VALUES (009, "989248912", 220,
2400.50, 5, 5);
```

```
/*
5 b) Agora crie um trigger que após o update na
conta_corrente atualiza essa conta.
Dica: Você pode utilizar os dados da tabela conta
corrente no insert do trigger utilizando
OLD.numerodacoluna e o saldo atualizado como
NEW.saldo)
*/
CREATE TRIGGER `conta_corrente_before_update` BEFORE UPDATE ON
`conta_corrente` FOR EACH ROW BEGIN
     SET NEW.saldo_atual = OLD.saldo_atual;
END



/*
6) Crie uma função que liste contas com saldo maior que o valor
informado.
*/
CREATE PROCEDURE `proc_maior_saldo`(
     IN `valor` DECIMAL(20,6),
     IN `tipo_conta` VARCHAR(10)
)
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
          case
               when tipo_conta = "cc" then
                    SELECT * FROM conta_corrente WHERE
conta_corrente.saldo_atual > valor;
               when tipo_conta = "cp" then
                    SELECT * FROM conta_poupanca WHERE
conta_poupanca.saldo > valor;
               when tipo_conta = "cred" then
                    SELECT * FROM conta_credito WHERE
conta_credito.fatura_atual > valor;
               ELSE SELECT 'Verifique valores de entrada';
          END case;
```

```
END

CALL proc_maior_saldo(10000, 'cp');


/*
7) Crie um procedimento que modifica o saldo de todas as
contas com o valor informado; Dica: Remova o safe updates
nas configs.
*/
CREATE PROCEDURE `proc_altera_saldo`(
     IN `valor` DECIMAL(20,6),
     IN `conta` VARCHAR(50)
)
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
     UPDATE conta_corrente SET conta_corrente.saldo_atual = valor
WHERE conta_corrente.numero_conta = conta;
     UPDATE conta_poupanca SET conta_poupanca.saldo = valor WHERE
conta_poupanca.numero_conta = conta;
     UPDATE conta_credito SET conta_credito.fatura_atual = valor;
END

/*
8) Agora crie os seguintes triggers:
a) Ao deletar uma conta apague também as
movimentações salvas (AFTER)
*/
CREATE TRIGGER `conta_corrente_after_delete` AFTER DELETE ON
`conta_corrente` FOR EACH ROW BEGIN
     DELETE FROM movimentacao
    WHERE movimentacao.fkcod_ccm = OLD.cod_cc;
END

DELETE FROM conta_corrente WHERE conta_corrente.cod_cc = 1;
```

```
/*
8 b) Ao criar uma conta salvar um registro na tabela
movimentação com saldo antigo e atual igual a 0
(AFTER)
*/
CREATE TRIGGER `conta_corrente_after_insert` AFTER INSERT ON
`conta_corrente` FOR EACH ROW BEGIN

     INSERT INTO movimentacao (numero_mov,
numero_conta,saldo_anterior, saldo_atual, fkcod_ccm, fkcod_cpm)
     VALUES (round(RAND()*100,2), NEW.numero_conta,  0.00,
NEW.saldo_atual, NEW.cod_cc, 5);

END

INSERT INTO conta_corrente (numero_conta,saldo_atual, nome, pix,
documento) VALUES ("9892492312", 850, "kaio", "00461306067",
"00461306067");



/*
8 c) Antes de cadastrar uma nova conta verifique se o saldo
informado é igual a 0, se não: atualize para 0 (BEFORE,
utilize IF ELSE)
*/
CREATE TRIGGER `check_insert_conta_corrente` BEFORE INSERT ON
`conta_corrente` FOR EACH ROW BEGIN

     if NEW.saldo_atual = 0 then
          INSERT INTO conta_corrente (numero_conta,saldo_atual,
nome, pix, documento)
               VALUES (NEW.numero_conta, NEW.saldo_atual,
NEW.nome, NEW.pix, NEW.documento);
     ELSE
          INSERT INTO conta_corrente (numero_conta,saldo_atual,
nome, pix, documento)
               VALUES (NEW.numero_conta, 0.00, NEW.nome, NEW.pix,
NEW.documento);
     END if;

END
```

```
/*
8 d) Antes de deletar uma conta, salvar um registro em
movimentação de que a conta está zerada (BEFORE)
*/
CREATE TRIGGER `check_valor_movimentacao` BEFORE DELETE ON
`conta_corrente` FOR EACH ROW BEGIN
     INSERT INTO movimentacao (numero_mov,
numero_conta,saldo_anterior, saldo_atual, fkcod_ccm, fkcod_cpm)
          VALUES (round(RAND()*100,2), old.numero_conta,
OLD.saldo_atual, 0.00, old.cod_cc, 5);
END
```