Код программы:

```
USE Factory;

DROP FUNCTION IF EXISTS GetOrderInnovice;
CREATE FUNCTION GetOrderInnovice(CustomerID_ int) RETURNS varchar(256)
    READS SQL DATA
BEGIN
    SET @Result = IFNULL((SELECT CONCAT('First order num. ',
                                        O.OrderID, ' with base cost ',
                                        O.OrderPrice, ' and service talon ',
                                        ST.TalonID, ' using products ',
                                        GROUP_CONCAT(DISTINCT P.ProductName SEPARATOR ', '), '.')
                          from Orders O
                                   JOIN ServiceTalons ST on O.OrderID = ST.OrderID
                                   JOIN Order2Product O2P on O.OrderID = O2P.OrderID
                                   JOIN Products P on O2P.ProductID = P.ProductID
                          WHERE O.CustomerID = CustomerID_
                            AND O.OrderEndDate > CURRENT_DATE()
                          GROUP BY O.OrderEndDate, O.OrderID
                          LIMIT 1), 'There are no such orders.');
    RETURN @Result;
end;

DROP FUNCTION IF EXISTS GetMinimalCost;
CREATE FUNCTION GetMinimalCost(OrderID_ int) RETURNS bigint
    READS SQL DATA
BEGIN
    SET @Result =
            (SELECT SUM(O2P.ProductAmount *
                        P2M.MaterialQuantity *
                        SMP.MaterialPrice)
             FROM Order2Product O2P
                      JOIN Product2Material P2M on O2P.ProductID = P2M.ProductID
                      JOIN Supplier2Material S2M on P2M.MaterialID = S2M.MaterialID
                      JOIN SupplierMaterialPrice SMP on S2M.MaterialPriceID = SMP.MaterialPriceID
             WHERE O2P.OrderID = OrderID_
               AND SMP.MaterialPrice
             GROUP BY O2P.OrderID
             HAVING MIN(SMP.MaterialPrice) > 0
            );
    RETURN @Result;
end;

DROP FUNCTION IF EXISTS PersonnelInfo;
CREATE FUNCTION PersonnelInfo(PersonalID_ int, Occupation_ varchar(64)) RETURNS VARCHAR(300)
    reads sql data
begin
    set @Result = (SELECT CONCAT(P.FirstName,
                                 IF(P.MiddleName is not null, CONCAT(' ', P.MiddleName), ''),
                                 ' ', P.LastName, ' working at ',
                                 D.DepartmentLocation, ' on ', COUNT(O.OrderID), ' orders and having ',
                                 IF(P.ManagerID IS NOT NULL AND (SELECT Occupation
                                                                 FROM Personal
                                                                 WHERE Personal.PersonalID = P.ManagerID
                                                                   and Personal.Occupation = Occupation_) IS NOT
NULL,
                                    (SELECT CONCAT(P2.FirstName,
                                                   IF(P2.MiddleName is not null, CONCAT(' ', P2.MiddleName), ''),
' ',
                                                   P2.LastName,
```

```sql
                                                      ' working at ',
                                                      D2.DepartmentLocation, ' on ', COUNT(O2.OrderID), ' orders')
                                          FROM Personal P2
                                                   JOIN Departments D2 on P2.DepartmentID = D2.DepartmentID
                                                   JOIN Order2Personal O2P2 on P2.PersonalID = O2P2.PersonalID
                                                   JOIN Orders O2 on O2P2.OrderID = O2.OrderID
                                          WHERE P2.PersonalID = P.ManagerID
                                            and P2.Occupation = Occupation_
                                          GROUP BY P2.FirstName),
                                         'noone'), ' as boss.')
                  FROM Personal P
                           JOIN Departments D on P.DepartmentID = D.DepartmentID
                           JOIN Order2Personal O2P on P.PersonalID = O2P.PersonalID
                           JOIN Orders O on O2P.OrderID = O.OrderID
                  WHERE P.PersonalID = PersonalID_
                    AND P.Occupation = Occupation_
                  GROUP BY P.FirstName);
    RETURN @Result;
end;

DROP PROCEDURE IF EXISTS ShowFullWorkingEnvironment;
CREATE PROCEDURE ShowFullWorkingEnvironment()
    MODIFIES SQL DATA
BEGIN
    DROP TABLE IF EXISTS temporal_product_analytic;
    CREATE temporary TABLE temporal_product_analytic
    (
        ProductID           int,
        ProductAmount       int,
        OrderID             int,
        AverageProductPrice int
    );
    INSERT INTO temporal_product_analytic
    SELECT P.ProductID,
           O2P.ProductAmount,
           O.OrderID,
           AVG(SMP.MaterialPrice * P2M.MaterialQuantity)
    FROM Products P
             JOIN Order2Product O2P on P.ProductID = O2P.ProductID
             JOIN Orders O on O2P.OrderID = O.OrderID
             JOIN Product2Material P2M on P.ProductID = P2M.ProductID
             JOIN Supplier2Material S2M on P2M.MaterialID = S2M.MaterialID
             JOIN SupplierMaterialPrice SMP on S2M.MaterialPriceID = SMP.MaterialPriceID
    GROUP BY P.ProductID, O2P.ProductAmount, O.OrderID;

    DROP TABLE IF EXISTS temporal_personal_analytic;
    CREATE temporary TABLE temporal_personal_analytic
    (
        PersonalID        int,
        MostFreqProductID int,
        OrderAmount       int
    );
    INSERT INTO temporal_personal_analytic
    SELECT P.PersonalID,
           PIDM.PID,
           COUNT(O2.OrderID)
    FROM Personal P
             JOIN Order2Personal O2P2 on P.PersonalID = O2P2.PersonalID
             JOIN Orders O2 on O2P2.OrderID = O2.OrderID
             JOIN Order2Product on O2.OrderID = Order2Product.OrderID
             JOIN (SELECT Order2Product.ProductID as PID, COUNT(*) as MFPID
                   FROM Order2Product
                   GROUP BY PID
                   ORDER BY MFPID DESC
                   LIMIT 1) PIDM on PIDM.PID = Order2Product.OrderID
    GROUP BY P.PersonalID, PIDM.PID;

    DROP TABLE IF EXISTS temporal_customer_analytic;
    CREATE temporary TABLE temporal_customer_analytic
    (
        CustomerID          int,
        OrderAmount         int,
        AverageOrderPrice   int,
        MostLoyalPersonalID int
    );
    INSERT INTO temporal_customer_analytic
    SELECT C.CustomerID,
           COUNT(O3.OrderID),
           AVG(O3.OrderPrice + S.MaterialPrice * O2P5.ProductAmount * M.MaterialQuantity),
           PMID.PID
    FROM Customers C
             JOIN Orders O3 on C.CustomerID = O3.CustomerID
             JOIN Order2Personal O2P4 on O3.OrderID = O2P4.OrderID
             JOIN Personal on O2P4.PersonalID = Personal.PersonalID
             JOIN Order2Product O2P5 on O3.OrderID = O2P5.OrderID
             JOIN Products P2 on O2P5.ProductID = P2.ProductID
             JOIN Product2Material M on P2.ProductID = M.ProductID
             JOIN Materials M2 on M.MaterialID = M2.MaterialID
             JOIN SupplierMaterialPrice S on M2.MaterialID = S.MaterialID
             JOIN (SELECT PID
                   FROM (SELECT Personal.PersonalID as PID, COUNT(*) as MFPID
                         FROM Personal
                         GROUP BY PID
                         ORDER BY MFPID DESC
                         LIMIT 1) as PM) as PMID
    group by C.CustomerID, PMID.PID
    having count(O3.OrderID) > 0;

    SELECT tca.*,
           tpea.*,
           tpa.*
    FROM temporal_customer_analytic tca
             JOIN Orders O4 on tca.CustomerID = O4.CustomerID
             JOIN temporal_personal_analytic tpea on tca.MostLoyalPersonalID = tpea.PersonalID
             JOIN temporal_product_analytic tpa
                  on tpea.MostFreqProductID = tpa.ProductID and tpa.OrderID = O4.OrderID;
end;

DROP PROCEDURE IF EXISTS ChangeCustomer;
CREATE PROCEDURE ChangeCustomer(OldCustomerID_ int, NewCustomerName_ varchar(64), NewCustomerPhone_ varchar(64))
    MODIFIES SQL DATA
BEGIN
    SET @QueryResult =
            (SELECT CONCAT(C.CustomerName, ' will be changed to ', NewCustomerName_, ', absorbing ',
                           IFNULL((SELECT count(OrderID)
                                   FROM Orders
                                   WHERE Orders.CustomerID = OldCustomerID_
                                     AND Orders.OrderEndDate < CURRENT_DATE()), 0),
                           ' orders, archiving ',
                           IFNULL((SELECT count(OrderID)
                                   FROM Orders
                                   WHERE Orders.CustomerID = OldCustomerID_
```

```
                                AND Orders.OrderEndDate >= CURRENT_DATE()), 0),
                      ' orders for a total cost of ',
                      IFNULL((SELECT SUM(Orders.OrderPrice)
                              FROM Orders
                              WHERE Orders.CustomerID = OldCustomerID_
                                and Orders.OrderEndDate >= CURRENT_DATE()), 0), '.')
         FROM Customers C
         WHERE C.CustomerID = OldCustomerID_);
SELECT @QueryResult as 'Changing the Customer';

INSERT INTO Customers (CustomerName, CustomerPhone) VALUE (NewCustomerName_, NewCustomerPhone_);
SET @NewCustomerID = LAST_INSERT_ID();

INSERT INTO ArchivedOrders (OriginalOrderID, OriginalCustomerID)
SELECT Orders.OrderID, OldCustomerID_
FROM Orders
WHERE CustomerID = OldCustomerID_
  and Orders.OrderEndDate >= CURRENT_DATE();

UPDATE Orders
SET Orders.CustomerID = @NewCustomerID
WHERE Orders.CustomerID = OldCustomerID_
  AND Orders.OrderEndDate < CURRENT_DATE();

DELETE
FROM OrderJournal2Product
WHERE JournalID in (SELECT JournalID
                    FROM OrderJournal
                    WHERE OrderID in (SELECT Orders.OrderID
                                      FROM Orders
                                      WHERE Orders.CustomerID = OldCustomerID_
                                        and Orders.OrderEndDate >= CURRENT_DATE()));

DELETE
FROM PreOrders
WHERE OrderID in (SELECT Orders.OrderID
                  FROM Orders
                  WHERE Orders.CustomerID = OldCustomerID_
                    and Orders.OrderEndDate >= CURRENT_DATE());

DELETE
FROM OrderJournal
WHERE OrderID in (SELECT Orders.OrderID
                  FROM Orders
                  WHERE Orders.CustomerID = OldCustomerID_
                    and Orders.OrderEndDate >= CURRENT_DATE());

DELETE
FROM ServiceTalons
WHERE ServiceTalons.OrderID in (SELECT Orders.OrderID
                                FROM Orders
                                WHERE Orders.CustomerID = OldCustomerID_
                                  and Orders.OrderEndDate >= CURRENT_DATE());

DELETE
FROM Order2Personal
WHERE Order2Personal.OrderID in (SELECT Orders.OrderID
                                 FROM Orders
                                 WHERE Orders.CustomerID = OldCustomerID_
                                   and Orders.OrderEndDate >= CURRENT_DATE());
```

```
    DELETE
    FROM Order2Product
    WHERE Order2Product.OrderID in (SELECT Orders.OrderID
                                    FROM Orders
                                    WHERE Orders.CustomerID = OldCustomerID_
                                      and Orders.OrderEndDate >= CURRENT_DATE());

    DELETE
    FROM Orders
    WHERE Orders.CustomerID = OldCustomerID_
      and Orders.OrderEndDate >= CURRENT_DATE();

    DELETE FROM Customers WHERE CustomerID = OldCustomerID_;
end;

DROP PROCEDURE IF EXISTS NewOrder;
CREATE PROCEDURE NewOrder(customerID_ INT,
                          orderPrice_ BIGINT,
                          orderStartDate_ DATE,
                          orderEndDate_ DATE,
                          technologyID_ INT,
                          personnelTable_ varchar(64),
                          productsTable_ varchar(64))
    MODIFIES SQL DATA
BEGIN
    INSERT INTO Orders (OrderPrice, OrderStartDate, OrderEndDate, CustomerID, TechnologyID)
        VALUE (orderPrice_, orderStartDate_, orderEndDate_, customerID_, technologyID_);
    SET @NewOrderID = LAST_INSERT_ID();

    INSERT INTO ServiceTalons (OrderID) VALUE (@NewOrderID);
    INSERT INTO PreOrders VALUE ((CONCAT(CAST(@NewOrderID as CHAR), '_', CAST(customerID_ as CHAR))),
                                 @NewOrderID);

    DROP TABLE IF EXISTS temp;
    CREATE temporary TABLE temp
    (
        PersonalID int
    );
    SET @queryPerosnal = CONCAT('INSERT INTO temp SELECT * FROM ', personnelTable_);
    PREPARE personalStatement FROM @queryPerosnal;
    EXECUTE personalStatement;
    DEALLOCATE PREPARE personalStatement;
    INSERT INTO Order2Personal
    SELECT *
    FROM (SELECT @NewOrderID) as NOI
             JOIN temp T on true;
    drop table temp;

    CREATE temporary TABLE temp
    (
        ProductID     int,
        ProductAmount int
    );
    SET @queryProduct = CONCAT('INSERT INTO temp SELECT * FROM ', productsTable_);
    PREPARE productStatement FROM @queryProduct;
    EXECUTE productStatement;
    DEALLOCATE PREPARE productStatement;
    # noinspection SqlInsertValues

    INSERT INTO Order2Product (OrderID, ProductID, ProductAmount)
    SELECT @NewOrderID, T.ProductID, T.ProductAmount
    from temp T;
    # inspection SqlInsertValues

    INSERT INTO OrderJournal (OrderID)
    SELECT @NewOrderID
    WHERE (SELECT (P3.ProductAmount - O2P3.ProductAmount)
           FROM Orders O2
                    JOIN Order2Product O2P3 on O2.OrderID = O2P3.OrderID
                    JOIN Products P3 on O2P3.ProductID = P3.ProductID
           WHERE O2.OrderID = @NewOrderID);

    INSERT INTO OrderJournal2Product (JournalID, ProductID, ProductAmount, SummaryDescription)
    SELECT OJ.JournalID,
           P.ProductID,
           ABS(O2P.ProductAmount - P.ProductAmount) as PDIF,
           CONCAT('To fullfil order num ', @NewOrderID,
                  ', we should build additional ',
                  ABS(O2P.ProductAmount - P.ProductAmount), ' ', P.ProductName, ' products.')
    FROM OrderJournal OJ
             JOIN Orders O on OJ.OrderID = O.OrderID and O.OrderID = @NewOrderID
             JOIN Order2Product O2P on O.OrderID = O2P.OrderID
             JOIN Products P on O2P.ProductID = P.ProductID
    WHERE P.ProductAmount - O2P.ProductAmount < 0;

    INSERT INTO SupplierJournal (SupplierID)
    SELECT S.SupplierID
    FROM Suppliers S
             JOIN Supplier2Material S2M2 on S.SupplierID = S2M2.SupplierID
             JOIN Product2Material P2M2 on S2M2.MaterialID = P2M2.MaterialID
             JOIN OrderJournal2Product OJ2P on P2M2.ProductID = OJ2P.ProductID
             JOIN OrderJournal J on OJ2P.JournalID = J.JournalID and J.OrderID = @NewOrderID;

    INSERT INTO SupplierJournal2Material (JournalID, MaterialID, MaterialAmount, TotalCost, SummaryDescription)
    SELECT SJ.JournalID,
           M.MaterialID,
           (P2M.MaterialQuantity * ABS(O2P2.ProductAmount - P2.ProductAmount)),
           MIN(SMP.MaterialPrice * P2M.MaterialQuantity * ABS(O2P2.ProductAmount - P2.ProductAmount)),
           CONCAT('To fullfil order num ', @NewOrderID,
                  ', we should order additional ',
                  P2M.MaterialQuantity * ABS(O2P2.ProductAmount - P2.ProductAmount),
                  ' ', M.MaterialName, ' materials to do ', P2.ProductName, ' for a price of ',
                  MIN(SMP.MaterialPrice * P2M.MaterialQuantity * ABS(O2P2.ProductAmount - P2.ProductAmount))) as
SumDesc
    FROM SupplierJournal SJ
             JOIN Suppliers S on SJ.SupplierID = S.SupplierID
             JOIN Supplier2Material S2M on S.SupplierID = S2M.SupplierID
             JOIN SupplierMaterialPrice SMP on S2M.MaterialPriceID = SMP.MaterialPriceID
             JOIN Materials M on S2M.MaterialID = M.MaterialID
             JOIN Product2Material P2M on M.MaterialID = P2M.MaterialID
             JOIN Products P2 on P2M.ProductID = P2.ProductID
             JOIN Order2Product O2P2 on O2P2.OrderID = @NewOrderID
    GROUP BY SJ.JournalID,
             M.MaterialID,
             P2M.MaterialQuantity,
             O2P2.ProductAmount,
             P2.ProductAmount,
             SMP.MaterialPrice,
             P2.ProductName;

    drop table temp;
end;
```

```
DROP PROCEDURE IF EXISTS DeleteOrder;
CREATE PROCEDURE DeleteOrder(OrderID_ int)
    MODIFIES SQL DATA
BEGIN
    DELETE FROM Order2Product WHERE OrderID = OrderID_;
    DELETE FROM Order2Personal WHERE OrderID = OrderID_;
    DELETE
    FROM OrderJournal2Product
    WHERE JournalID in
          (SELECT JournalID FROM OrderJournal WHERE OrderJournal.OrderID = OrderID_);
    DELETE FROM ServiceTalons WHERE OrderID = OrderID_;
    DELETE FROM OrderJournal WHERE OrderID = OrderID_;
    DELETE FROM PreOrders WHERE OrderID = OrderID_;
    DELETE FROM Orders WHERE OrderID = OrderID_;
end;

DROP PROCEDURE IF EXISTS UpdateOrder;
CREATE PROCEDURE UpdateOrder(OrderID_ int,
                             OrderPrice_ int,
                             OrderStartDate_ DATE,
                             OrderEndDate_ DATE,
                             TechnologyID_ int)
    MODIFIES SQL DATA
BEGIN
    UPDATE Orders
    SET OrderPrice     = OrderPrice_,
        OrderStartDate = OrderStartDate_,
        OrderEndDate   = OrderEndDate_,
        TechnologyID   = TechnologyID_
    WHERE OrderID = OrderID_;
end;

DROP PROCEDURE IF EXISTS GetAveragePricesMaterial;
CREATE PROCEDURE GetAveragePricesMaterial(MaterialID_ int)
    MODIFIES SQL DATA
BEGIN
    SELECT M.MaterialID as 'MaterialID', AVG(SMP.MaterialPrice) as 'Average Price'
    FROM Materials M
             JOIN Supplier2Material S2M ON M.MaterialID = S2M.MaterialID
             JOIN SupplierMaterialPrice SMP on S2M.MaterialPriceID = SMP.MaterialPriceID
    WHERE M.MaterialID = MaterialID_
    GROUP BY M.MaterialID;
end;
```