Домашнее задание № 4. Запросы.

Задание

Требуется составить, отладить и проверить 23 инструкции обработки данных, руководствуясь примерами ниже.

Указания

Запросы желательно строить для разработанной вами в ДЗ № 3 базы данных. Допускается использовать другие базы данных, например Adventure Works

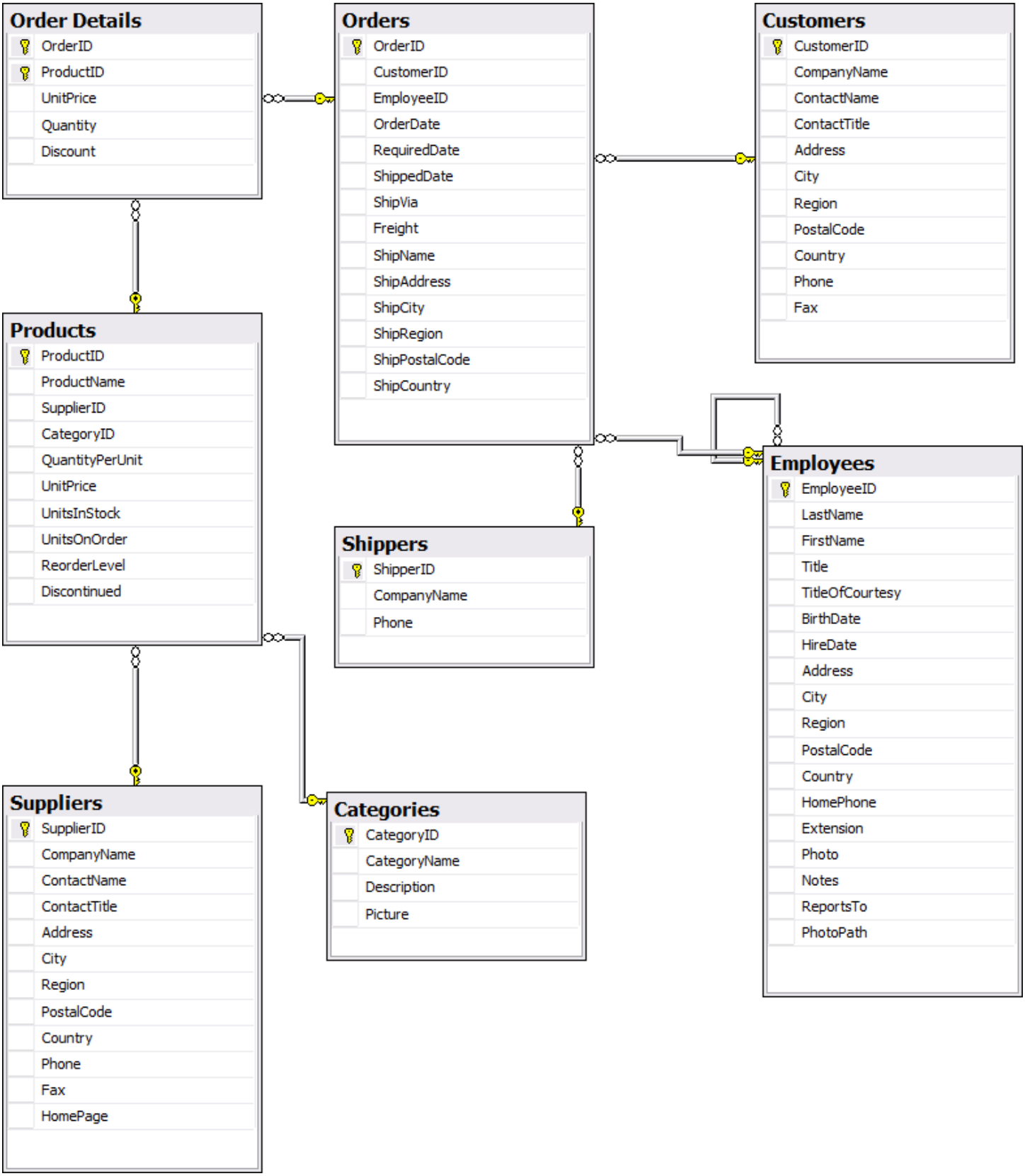
Для составления запроса с использованием рекурсивного обобщенного табличного выражения потребуется таблица, содержащая внешний ключ, ссылающийся на эту же таблицу. Если такой таблицы нет в вашей БД, то создайте её для этого примера.

Форма отчёта

В качестве отчёта требуется предоставить сценарии запросов на SQL для конкретной СУБД.

Примеры запросов.

Схема базы данных



1. Инструкция SELECT, использующая предикат сравнения.

```
SELECT DISTINCT C1.City, C1.CompanyName
FROM Customers C1 JOIN Customers AS C2 ON C2.City = C1.City
WHERE C2.CustomerID <> C1.CustomerID AND C1.Country = 'Argentina'
ORDER BY C1.City, C1.CompanyName
```

2. Инструкция SELECT, использующая предикат BETWEEN.

```sql
-- Получить список клиентов, сделавших заказы между '1997-01-01' и '1997-03-31'
SELECT DISTINCT CustomerID, OrderDate
FROM Orders
WHERE OrderDate BETWEEN '1997-01-01' AND '1997-03-31'
```

3. Инструкция SELECT, использующая предикат LIKE.

```sql
-- Получить список категорий продуктов в описании которых присутствует слово 'pasta'
SELECT DISTINCT CategoryName
FROM Categories JOIN Products ON Products.categoryID = Categories.CategoryID
WHERE Description LIKE '%pasta%'
```

4. Инструкция SELECT, использующая предикат IN с вложенным подзапросом.

```sql
-- Получить список заказов для клиентов из Лондона, оформленных через сотрудника 1
SELECT OrderID, CustomerID, EmployeeID, OrderDate
FROM Orders
WHERE CustomerID IN
(
              SELECT CustomerID
                FROM Customers
                WHERE City = 'London'
) AND EmployeeID = 1
```

5. Инструкция SELECT, использующая предикат EXISTS с вложенным подзапросом.

```sql
-- Получить список продуктов, которые никто никогда не заказывал
SELECT ProductID, ProductName
FROM Products
WHERE EXISTS
(
SELECT Products.ProductID
                FROM Products LEFT OUTER JOIN [Order Details]
ON Products.ProductID = [Order Details].ProductID
                WHERE [Order Details].ProductID IS NULL
)
```

6. Инструкция SELECT, использующая предикат сравнения с квантором.

```sql
-- Получить список продуктов, цена которых больше цены любого продукта категории 2
SELECT ProductID, ProductName, UnitPrice
FROM Products
WHERE UnitPrice > ALL
(
SELECT UnitPrice
                FROM Products
                WHERE CategoryID = 2
)
```

7. Инструкция SELECT, использующая агрегатные функции в выражениях столбцов.

```sql
SELECT    AVG(TotalPrice) AS 'Actual AVG',
    SUM(TotalPrice) / COUNT(OrderID) AS 'Calc AVG'
FROM (
SELECT OrderID, SUM(UnitPrice*Quantity*(1-Discount)) AS TotalPrice
    FROM [Order Details]
    GROUP BY OrderID
     )    AS TotOrders
```

8. Инструкция SELECT, использующая скалярные подзапросы в выражениях столбцов.

```
SELECT    ProductID,
UnitPrice,
    (
        SELECT AVG(UnitPrice)
        FROM [Order Details]
        WHERE [Order Details].ProductID = Products.ProductID
) AS AvgPrice,
(
        SELECT MIN(UnitPrice)
        FROM [Order Details]
        WHERE [Order Details].ProductID = Products.ProductID
) AS MaxPrice,
ProductName
FROM Products
WHERE CategoryID = 1
```

9. Инструкция SELECT, использующая простое выражение CASE.

```
SELECT CompanyName, OrderID,
    CASE YEAR(OrderDate)
WHEN YEAR(Getdate()) THEN 'This Year'
WHEN YEAR(GetDate()) - 1 THEN 'Last year'
ELSE CAST(DATEDIFF(year, OrderDate, Getdate()) AS varchar(5)) + ' years ago'
    END AS 'When'
FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID
```

10. Инструкция SELECT, использующая поисковое выражение CASE.

```
SELECT ProductName,
    CASE
        WHEN UnitPrice < 10 THEN 'Inexpensive'
        WHEN UnitPrice < 50 THEN 'Fair'
        WHEN UNitPrice < 100 THEN 'Expensive'
        ELSE 'Very Expensive'
    END AS Price
FROM products
```

11. Создание новой временной локальной таблицы из результирующего набора данных инструкции SELECT.

```
SELECT ProductID,
       SUM(Quantity) AS SQ,
       CAST(SUM(UnitPrice*Quantity*(1.0-Discount))AS money) AS SR
INTO #BestSelling
FROM [Order Details]
WHERE ProductID IS NOT NULL
GROUP BY productID
```

12. Инструкция SELECT, использующая вложенные коррелированные подзапросы в качестве производных таблиц в предложении FROM.

```
SELECT 'By units' AS Criteria, ProductName as 'Best Selling'
FROM Products P JOIN
    (
SELECT TOP 1 ProductID, SUM(Quantity) AS SQ
                FROM [Order Details]
                GROUP BY productID
                ORDER BY SQ DESC
```

```
    ) AS OD ON OD.ProductID = P.ProductID
UNION
SELECT 'By revenue' AS Criteria, ProductName as 'Best Selling'
FROM Products P JOIN
    (
        SELECT TOP 1 ProductID, SUM(UnitPrice*Quantity*(1-Discount)) AS SR
                FROM [Order Details]
                GROUP BY ProductID
                ORDER BY SR DESC
    ) AS OD ON OD.ProductID = P.ProductID
```

13. Инструкция SELECT, использующая вложенные подзапросы с уровнем вложенности 3.

```
SELECT 'By units' AS Criteria, ProductName as 'Best Selling'
FROM Products
WHERE ProductID =
    (
        SELECT ProductID
        FROM [Order Details]
        GROUP BY ProductID
        HAVING SUM(Quantity) =
            (
                SELECT MAX(SQ)
                        FROM
                    (
                        SELECT SUM(Quantity) as SQ
                                    FROM [Order Details]
                                    GROUP BY ProductID
                    ) AS OD
            )
    )
UNION
SELECT 'By revenue' AS Criteria, ProductName as 'Best Selling'
FROM Products
WHERE ProductID =
    (
        SELECT ProductID
        FROM [Order Details]
        GROUP BY ProductID
        HAVING SUM(UnitPrice*Quantity*(1-Discount)) =
            (
                SELECT MAX(SQ)
                FROM
                    (
                        SELECT SUM(UnitPrice*Quantity*(1-Discount)) AS SQ
                        FROM [Order Details]
                        GROUP BY ProductID
                    ) AS OD
            )
)
```

14. Инструкция SELECT, консолидирующая данные с помощью предложения GROUP BY, но без предложения HAVING.

```
-- Для каждого заказанного продукта категории 1 получить его цену, среднюю цену,
-- минимальную цену и название продукта
SELECT P.ProductID,
     P.UnitPrice,
     AVG(OD.UnitPrice) AS AvgPrice,
     MIN(OD.UnitPrice) AS MinPrice,
```

```
     P.ProductName
FROM Products P LEFT OUTER JOIN [Order Details] OD ON OD.ProductID = P.ProductID
WHERE CategoryID = 1
GROUP BY P.productID, P.UnitPrice, P.ProductName
```

15. Инструкция SELECT, консолидирующая данные с помощью предложения GROUP BY и предложения HAVING.

```
-- Получить список категорий продуктов, средняя цена которых больше общей средней
-- цены продуктов
SELECT CategoryID, AVG(UnitPrice) AS 'Average Price'
FROM Products P
GROUP BY CategoryID
HAVING AVG(UnitPrice) >
(
    SELECT AVG(UnitPrice) AS MPrice
    FROM Products
)
```

16. Однострочная инструкция INSERT, выполняющая вставку в таблицу одной строки значений.

```
INSERT Products (ProductName, SupplierID, CategoryID, QuantityPerUnit, ReorderLevel, Discontinued)
VALUES ('Donut', NULL, NULL, '6 pieces', DEFAULT, DEFAULT)
```

17. Многострочная инструкция INSERT, выполняющая вставку в таблицу результирующего набора данных вложенного подзапроса.

```
INSERT [Order Details] (OrderID, ProductID, Unitprice, Quantity, Discount)
SELECT (
        SELECT MAX(OrderID)
        FROM Orders
        WHERE CustomerID = 'ALFKI'
    ), ProductID, UnitPrice, 10, 0.1
FROM Products
WHERE ProductName = 'Tofu'
```

18. Простая инструкция UPDATE.

```
UPDATE Products
SET UnitPrice = UnitPrice * 1.5
WHERE ProductID = 35
```

19. Инструкция UPDATE со скалярным подзапросом в предложении SET.

```
UPDATE Products
SET UnitPrice =
(
    SELECT AVG(UnitPrice)
        FROM [Order Details]
        WHERE ProductID = 37
)
WHERE ProductID = 37
```

20. Простая инструкция DELETE.

```
DELETE Orders
WHERE CustomerID IS NULL
```

21. Инструкция DELETE с вложенным коррелированным подзапросом в предложении WHERE.

```sql
-- Пример для базы данных AdventureWorks
DELETE FROM Production.Product
WHERE ProductID IN
(
    SELECT Product.ProductID
    FROM Production.Product LEFT OUTER JOIN Sales.SalesOrderDetail
        ON Product.ProductID = SalesOrderDetail.ProductID
    WHERE SalesOrderDetail.ProductID IS NULL
        AND Product.ProductSubCategoryID = 5
)
```

22. Инструкция SELECT, использующая простое обобщенное табличное выражение.

```sql
-- Пример для базы данных SPJ
WITH CTE (SupplierNo, NumberOfShips)
AS
(
    SELECT Sno, COUNT(*) AS Total
    FROM SPJ
    WHERE Sno IS NOT NULL
    GROUP BY Sno
)
SELECT AVG(NumberOfShips) AS 'Среднее количество поставок для поставщиков'
FROM CTE
```

23. Инструкция SELECT, использующая рекурсивное обобщенное табличное выражение.

```sql
-- Создание таблицы.
CREATE TABLE dbo.MyEmployees
(
    EmployeeID smallint NOT NULL,
    FirstName nvarchar(30)  NOT NULL,
    LastName  nvarchar(40) NOT NULL,
    Title nvarchar(50) NOT NULL,
    DeptID smallint NOT NULL,
    ManagerID int NULL,
     CONSTRAINT PK_EmployeeID PRIMARY KEY CLUSTERED (EmployeeID ASC)
);
GO
-- Заполнение таблицы значениями.
INSERT INTO dbo.MyEmployees VALUES (1, N'Иван', N'Петров', N'Главный исполнительный директор',16,NULL) ;
…
GO
-- Определение ОТВ
WITH DirectReports (ManagerID, EmployeeID, Title, DeptID, Level)
AS
(
-- Определение закрепленного элемента
    SELECT e.ManagerID, e.EmployeeID, e.Title, e.DeptID, 0 AS Level
    FROM dbo.MyEmployees AS e
    WHERE ManagerID IS NULL
    UNION ALL
-- Определение рекурсивного элемента
    SELECT e.ManagerID, e.EmployeeID, e.Title, e.DeptID, Level + 1
    FROM dbo.MyEmployees AS e INNER JOIN DirectReports AS d ON e.ManagerID = d.EmployeeID
)
-- Инструкция, использующая ОТВ
SELECT ManagerID, EmployeeID, Title, DeptID, Level
FROM DirectReports ;
```