Практическая работа 6

Цель работы: Использование оконных функций и предложения OVER.

Предложение OVER определяет секционирование и упорядочение набора строк до применения соответствующей оконной функции.

```
OVER (
 [ <PARTITION BY столбец> ]
 [ <ORDER BY столбец> ]
 [ <ROW или RANGE столбец> ]
)
```

PARTITION BY разделяет результирующий набор запроса на секции. Оконная функция применяется к каждой секции отдельно, и вычисление начинается заново для каждой секции.

ORDER BY определяет логический порядок строк в каждой секции результирующего набора.

ROW or RANGE ограничивает строки в пределах секции, указывая начальную и конечную точки.

Параметр CURRENT ROW указывает, что окно начинается или заканчивается на текущей строке при использовании совместно с предложением ROWS или что окно заканчивается на текущем значении при использовании с предложением RANGE. CURRENT ROW может быть задана и как начальная, и как конечная точка.

Параметр BETWEEN <граница рамки окна > AND <граница рамки окна > используется совместно с предложением ROWS или RANGE для указания нижней (начальной) или верхней (конечной) граничной точки окна. <граница рамки окна> определяет граничную начальную точку, а <граница рамки окна> определяет граничную конечную точку.

Параметр UNBOUNDED FOLLOWING указывает, что окно заканчивается на последней строке секции. UNBOUNDED FOLLOWING может быть указано только как конечная точка окна.

Пример:

```
RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
```

В данном случае параметр определяет, что окно начинается на текущей строке и заканчивается на последней строке секции.

```
ROWS BETWEEN 2 FOLLOWING AND 10 FOLLOWIN
```

В этом примере параметр определяет, что окно начинается на второй строке после текущей и заканчивается на десятой строке после текущей строки. Эта спецификация не допускается в предложении RANGE.

Фактически предложение OVER виртуально разбивает выбранные строки на наборы, окна, определяемые в условии PARTITION BY, упорядочивает эти строки по столбцам определенным ORDER BY. В рамках этих наборов, окон, выполняются те или иные агрегирующие, статистические и иные функции. Результат выполнения этих функций формирует отдельный столбец с одинаковыми значениями для каждой строки в наборе, окне. Однако можно для каждой строки в наборе формировать свое значение функции, для чего используют параметр ROW или RANGE, который определяет диапазон строк, в наборе, окне, с которыми будет работать функция.

```
SELECT SalesORDERID, ProductID, ORDERQty ,SUM(ORDERQty) OVER(PARTITION BY SalesORDERID) AS Total

,AVG(ORDERQty) OVER(PARTITION BY SalesORDERID) AS "Avg"

,COUNT(ORDERQty) OVER(PARTITION BY SalesORDERID) AS "Count"

,MIN(ORDERQty) OVER(PARTITION BY SalesORDERID) AS "MIN"

,MAX(ORDERQty) OVER(PARTITION BY SalesORDERID) AS "Max"

FROM Sales.SalesORDERDetail

WHERE SalesORDERID IN(43659,43664)
```

Следующий запрос использует приведение типа.

```
SELECT SalesORDERID, ProductID, ORDERQty

,SUM(ORDERQty) OVER(PARTITION BY SalesORDERID) AS Total

,CAST(1. * ORDERQty / SUM(ORDERQty) OVER(PARTITION BY SalesORDERID)

*100 AS DECIMAL(5,2))AS "Percent BY ProductID"

FROM Sales.SalesORDERDetail

WHERE SalesORDERID IN(43659,43664)
```

Можно выполнять функцию не на всем наборе, но на части набора. Часть определяется в зависимости от строки и ее положения в наборе.

```
SELECT BusINessEntityID, TerritoryID

,CONVERT(varchar(20),SalesYTD,1) AS  SalesYTD

,DATEPART(yy,ModifiedDate) AS SalesYear

,CONVERT(varchar(20),SUM(SalesYTD) OVER (PARTITION BY TerritoryID

ORDER BY DATEPART(yy,ModifiedDate)

ROWS BETWEEN CURRENT ROW AND 1 FOLLOWING ),1) AS CumulativeTotal

FROM Sales.SalesPerson

WHERE TerritoryID IS NULL OR TerritoryID < 5
```

Аналитические функции.

Функция FIRST_VALUE возвращает первое значение из упорядоченного набора значений.

Синтаксис

```
FIRST_VALUE ( [scalar_expressiON ] ) OVER ( [ partitiON_BY_clause ] ORDER_BY_clause [ rows_range_clause ] )
```

scalar_expressiON может быть столбцом, вложенным запросом.

Получение имени самого дешевого продукта в заданной категории продуктов.

```
SELECT Name, ListPrice,

FIRST_VALUE(Name) OVER (ORDER BY ListPrice ASC) AS LeAStExpensive

FROM Production.Product

WHERE ProductSubcategoryID = 37
```

Функция LAST_VALUE возвращает последнее значение из упорядоченного набора значений.

Синтаксис

```
LAST_VALUE ( [ scalar_expressiON ] ) OVER ( [ partitiON_BY_clause ] ORDER_BY_clause rows_range_clause )
```

scalar_expressiON может быть столбцом, вложенным запросом.

Возвращение даты найма последнего сотрудника каждого отдела для указанной заработной платы (Rate). Предложение PARTITION BY разделяет сотрудников по отделам, а функция LAST_VALUE применяется к каждой секции в отдельности. Предложение ORDER BY, указанное в предложении OVER, определяет логический порядок, в котором функция LAST_VALUE применяется к строкам каждой секции.

```
SELECT Department, LastName, Rate, HireDate,

LAST_VALUE(HireDate) OVER (PARTITION BY Department ORDER BY Rate) AS LAStValue

FROM HumanResources.vEmployeeDepartmentHistory AS edh

INNER JOIN HumanResources.EmployeePayHistory AS eph

ON eph.BusINessEntityID = edh.BusINessEntityID

INNER JOIN HumanResources.Employee AS e

ON e.BusINessEntityID = edh.BusINessEntityID

WHERE Department IN (N'INformatiON Services',N'Document CONtrol')
```

Функция LAG обеспечивает доступ к строке с заданным физическим смещением перед началом текущей строки.

```
LAG (scalar_expressiON [,offset] [,default])

OVER ( [ partitiON_BY_clause ] ORDER_BY_clause )
```

scalar_expressiON – возвращаемое значение основано на указанном смещении.

offset – количество строк до строки перед текущей строкой, из которой необходимо получить значение.

default – возвращаемое значение, когда offset находится за пределами секции.

Нахождение квоты для работника за год и предыдущий год.

Квоты менялись несколько раз в год, но для первой установленной в году квоты нет предыдущего значения от 2010 года, их не включают в выборку.

```
SELECT BusINessEntityID, YEAR(QuotaDate) AS SalesYear, SalesQuota AS CurrentQuota,

LAG(SalesQuota, 1,0) OVER (ORDER BY YEAR(QuotaDate)) AS PreviousQuota

FROM Sales.SalesPersonQuotaHistory

WHERE BusINessEntityID = 275 AND YEAR(QuotaDate) IN ('2011','2012')
```

Функция LEAD – доступ к строке на заданном физическом смещении после текущей строки.

Синтаксис

```
LEAD ( scalar_expressiON [ ,offset ] , [ default ] ) OVER ( [ partitiON_BY_clause ] ORDER_BY_clause )
```

Пример, получение квот продаж для указанного работника за последующие годы.

```
SELECT BusINessEntityID, YEAR(QuotaDate) AS SalesYear, SalesQuota AS CurrentQuota,

LEAD(SalesQuota, 1,0) OVER (ORDER BY YEAR(QuotaDate)) AS NextQuota

FROM Sales.SalesPersonQuotaHistory

WHERE BusINessEntityID = 275 AND YEAR(QuotaDate) IN ('2011','2012')
```

Функция NTILE распределяет строки упорядоченной секции в заданное количество групп. Группы нумеруются, начиная с единицы. Для каждой строки функция NTILE возвращает номер группы, которой принадлежит строка.

Синтаксис

NTILE (INteger_expressiON) OVER ([<partitiON_BY_clause>] < ORDER_BY_clause >)

INteger_expressiON – положительное целое выражение, указывающее число групп, на которые необходимо разделить каждую секцию.

Функция ROW_NUMBER нумерует выходные данные результирующего набора.

```
ROW_NUMBER ( )
OVER ( [ PARTITION BY value_expressiON , ... [ n ] ] ORDER_BY_clause )
```

Пример использования функций.

```
SELECT p.FirstName, p.LastName
,ROW_NUMBER() OVER (ORDER BY a.PostalCode) AS "Row Number"
,NTILE(4) OVER (ORDER BY a.PostalCode) AS Quartile
,s.SalesYTD
,a.PostalCode
FROM Sales.SalesPerson AS s
INNER JOIN Person.Person AS p
ON s.BusINessEntityID = p.BusINessEntityID
INNER JOIN Person.Address AS a
ON a.AddressID = p.BusINessEntityID
WHERE TerritoryID IS NOT NULL AND SalesYTD <> 0
```

Примеры запросов с решениями

1 Найти долю затрат каждого покупателя на каждый купленный им продукт среди общих его затрат в данной сети магазинов. Можно использовать обобщенное табличное выражение.

```
SELECT [SalesORDERID], p.[ProductID],
[ProductSubcategoryID],
[ORDERQty]*[UnitPrice],
[ORDERQty]*[UnitPrice]/sum([ORDERQty]*[UnitPrice])
OVER(partitiON BY [SalesORDERID]
, [ProductSubcategoryID])
FROM [Sales].[SalesORDERDetail] AS SOD INner JOIN
[Production].[Product] AS p
ON SOD.ProductID=p.ProductID
```

2 Для одного выбранного покупателя вывести, для каждой покупки (чека), разницу в деньгах между этой и следующей покупкой.

Вариант 1

```
with tmp (customer, ORDERid, total)  AS
```

```
(SELECT soh.CustomerID, soh.SalesORDERID,
sum(sod.[ORDERQty]*[UnitPrice]) AS total
FROM [Sales].[SalesORDERHeader] AS SOH INner JOIN
[Sales].[SalesORDERDetail] AS SOD
ON soh.SalesORDERID=sod.SalesORDERID
GROUP BY soh.CustomerID, soh.SalesORDERID)
SELECT customer, ORDERid, total,
total-LEAD(total,1,0)
OVER(partitiON BY customer ORDER BY ORDERid)
FROM tmp
```

3 Вывести следующую информацию: номер покупателя, номер чека этого покупателя, отсортированные по покупателям, номерам чека (по возрастанию). Третья колонка должна содержать в каждой своей строке сумму текущего чека покупателя со всеми предыдущими чеками этого покупателя.

Вариант 1

```
with tmp (cus, ord, ORDERsum)
AS (SELECT oh.CustomerID, od.SalesORDERID,
sum(od.[UnitPrice]*[ORDERQty])
FROM [Sales].[SalesORDERDetail] AS OD
INner JOIN
[Sales].[SalesORDERHeader] AS OH
ON od.SalesORDERID=oh.SalesORDERID
GROUP BY oh.CustomerID, od.SalesORDERID)
SELECT
cus, ord, ORDERsum,
sum(ORDERsum)
OVER(partitiON BY cus ORDER BY ord desc
RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING)
FROM tmp
```

Задания для самостоятельной работы

1 Найти долю продаж каждого продукта (цена продукта * количество продукта), на каждый чек, в денежном выражении.

2 Вывести на экран список продуктов, их стоимость, а также разницу между стоимостью этого продукта и стоимостью самого дешевого продукта в той же подкатегории, к которой относится продукт.

3 Вывести три колонки: номер покупателя, номер чека покупателя (отсортированный по возрастанию даты чека) и искусственно введенный порядковый номер текущего чека, начиная с 1, для каждого покупателя.

4 Вывести номера продуктов, таких что их цена выше средней цены продукта в подкатегории, к которой относится продукт. Запрос реализовать двумя способами. В одном из решений допускается использование обобщенного табличного выражения.

5 Вывести на экран номер продукта, название продукта, а также информацию о среднем количестве этого продукта, приходящихся на три последних по дате чека, в которых был этот продукт.