### Определения (Лекция 14)

**Aggregate window functions и Partition:**

|  |  |
| --- | --- |
| DECLARE @T TABLE (  id int, val float)  INSERT INTO @T  VALUES  (1, 2),  (1, 3),  (2, 4)  Можно использовать count, max, min… | SELECT \*,  ROW\_NUMBER() OVER(PARTITION BY id ORDER BY val ASC) as НомГр,  ROW\_NUMBER() OVER(ORDER BY val ASC) as НомТаб,  SUM(val) OVER(PARTITION BY id) СмГр, SUM(val) OVER() СмВc,  AVG(val) OVER(PARTITION BY id) СрГр, AVG(val) OVER() СрВс  FROM @T |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | id | val | НомГр | НомТаб | СмГр | СмВc | СрГр | СрВс | | 1 | 2 | 1 | 1 | 5 | 9 | 2.5 | 3 | | 1 | 3 | 2 | 2 | 5 | 9 | 2.5 | 3 | | 2 | 4 | 1 | 3 | 4 | 9 | 4 | 3 | |

**Нововведения с MS SQL 2012**

|  |  |  |  |
| --- | --- | --- | --- |
| DECLARE @T TABLE (id int) INSERT INTO @T VALUES (1), (2), (4), (5), (6) | | | |
| SELECT id,  SUM(id) OVER(  ORDER BY id  **ROWS** BETWEEN  UNBOUNDED PRECEDING AND CURRENT ROW ) AS runid  FROM @T | |  |  | | --- | --- | | **id** | **runid** | | 1 | 1 | | 2 | 3 | | 4 | 7 | | 5 | 12 | | 6 | 18 | | SELECT id,  SUM(id) OVER(  ORDER BY id  **ROWS** BETWEEN 2 PRECEDING AND  1 FOLLOWING) AS runid  FROM @T | |  |  | | --- | --- | | **id** | **runid** | | 1 | 3 | | 2 | 7 | | 4 | 12 | | 5 | 17 | | 6 | 15 | |

**Offset window functions**

|  |  |  |  |
| --- | --- | --- | --- |
| DECLARE @T TABLE (id int, val int) INSERT INTO @T VALUES (1, 1), (1, 5), (2, 2), (2, 6), (4, 1) | | | |
| SELECT id,  **FIRST\_VALUE**(val) OVER(ORDER BY val) f1, **LAST\_VALUE** (val) OVER(ORDER BY val) l1  FROM @T  ORDER BY id | |  |  |  | | --- | --- | --- | | **id** | **f1** | **l1** | | 1 | 1 | 1 | | 1 | 1 | 5 | | 2 | 1 | 2 | | 2 | 1 | 6 | | 4 | 1 | 1 |   Зачем нужны FIRST и LAST, если есть MIN и MAX? | SELECT id,  FIRST\_VALUE(val)  OVER(ORDER BY val  **ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW**) f1,  LAST\_VALUE (val)  OVER(ORDER BY val  **ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING**) l1  FROM @T  ORDER BY id | |  |  |  | | --- | --- | --- | | **id** | **f1** | **l1** | | 1 | 1 | 6 | | 1 | 1 | 6 | | 2 | 1 | 6 | | 2 | 1 | 6 | | 4 | 1 | 6 |   Не забывайте, что PARTITION здесь так же можно использовать. |
| SELECT id, val,  **LAG**(val, 2, 10)  OVER(ORDER BY id) lg,  **LEAD**(val)  OVER(ORDER BY id) ld  FROM @T | |  |  |  |  | | --- | --- | --- | --- | | **id** | **val** | **lg** | **ld** | | 1 | 1 | 10 | 5 | | 1 | 5 | 10 | 2 | | 2 | 2 | 1 | 6 | | 2 | 6 | 5 | 1 | | 4 | 1 | 2 | NULL | | SELECT id, val,  **LEAD**(val)  OVER (PARTITION BY id  ORDER BY val) ld  FROM @T | |  |  |  | | --- | --- | --- | | **id** | **val** | **ld** | | 1 | 1 | 5 | | 1 | 5 | NULL | | 2 | 2 | 6 | | 2 | 6 | NULL | | 4 | 1 | NULL | |

# Процедуры, Функции и View:

|  |  |  |
| --- | --- | --- |
| CREATE PROCEDURE  [dbo].[Товары\_по\_массе]  AS  SELECT Товар\_ID  FROM Товары  WHERE Масса > 100  GO  EXEC Товары\_по\_массе | CREATE PROCEDURE  [dbo].[Товары\_по\_массе\_параметр]  @param int  AS  SELECT Товар\_ID  FROM Товары  WHERE Масса > @param  GO  EXEC Товары\_по\_масса\_параметр 100 | CREATE TABLE #T(Товар\_ID int)  INSERT INTO #T  EXEC Товары\_по\_масса\_параметр 100  SELECT \*  FROM #T  DROP TABLE #T |

Еще пример помещения результата процедуры в таблицу

|  |  |
| --- | --- |
| ALTER PROCEDURE [dbo].[Товары\_по\_массе\_параметр]  @param int  AS  INSERT INTO #T  SELECT Товар\_ID  FROM Товары  WHERE Масса > @param  GO | CREATE TABLE #T (Товар\_ID int)  EXEC Товары\_по\_массе\_параметр 100  SELECT \* FROM #T  DROP TABLE #T |

Процедура с дефолтными и выходными параметрами:

|  |  |
| --- | --- |
| CREATE PROC Колво\_товаров\_по\_массе\_деф\_параметр  @n AS INT,  @m AS float = 50,  @num AS INT OUTPUT  AS  SET @num =  (  SELECT COUNT(\*)  FROM Товары  WHERE Масса < @m AND Товар\_ID > @n  )  GO | DECLARE @count AS INT;  EXEC Колво\_товаров\_по\_массе\_деф\_параметр  @n = 1,  --@m = 10,  @num = @count OUTPUT;  SELECT @count AS numgoods;  Если оставить закомментированным значение параметра @m, то в процедуре будет использоваться дефолтное значение. |

**Функции:**

|  |  |  |
| --- | --- | --- |
| CREATE FUNCTION  dbo.f\_Товары\_по\_массе\_число()  RETURNS FLOAT AS  BEGIN  RETURN  (  SELECT MAX(Масса)  FROM Товары  )  END  GO  SELECT dbo.f\_Товары\_по\_массе\_число() | CREATE FUNCTION  dbo.f\_Товары\_по\_массе()  RETURNS @Товары TABLE (Товар\_ID int)  AS  BEGIN  INSERT INTO @Товары  SELECT Товар\_ID  FROM Товары  WHERE Масса > 100  RETURN  END  GO  SELECT \*  FROM dbo.f\_Товары\_по\_массе() | CREATE FUNCTION  dbo.f\_Товары\_по\_массе\_параметр(@m int)  RETURNS @Товары TABLE (Товар\_ID int)  AS  BEGIN  INSERT INTO @Товары  SELECT Товар\_ID  FROM Товары  WHERE Масса > **@m**  RETURN  END  GO  SELECT \*  FROM f\_Товары\_по\_массе\_параметр (10) |

Еще примеры:

|  |  |  |
| --- | --- | --- |
| (Inline Table-Valued Functions):  CREATE FUNCTION  dbo.f\_Товары\_по\_массе\_параметр(@m int)  RETURNS TABLE  AS  RETURN  SELECT Товар\_ID  FROM Товары  WHERE Масса > @m  GO  SELECT \*  FROM f\_Товары\_по\_массе\_параметр (10) | (Views)  CREATE VIEW  dbo.v\_Товары\_по\_массе  AS  SELECT Товар\_ID  FROM Товары  WHERE Масса > 100  GO  SELECT \*  FROM dbo.v\_Товары\_по\_массе | System views: sys.objects  IF OBJECT\_ID  (N'dbo.f\_test', N'FN') IS NOT NULL  DROP FUNCTION dbo.f\_test;  GO  SELECT OBJECT\_ID('tempdb..#mytemptable') |

**Оператор Apply (**применяет выражение справа от оператора к каждой строке выражения слева от оператора**):**

|  |  |
| --- | --- |
| SELECT \*  FROM Товары CROSS JOIN Склады | SELECT \*  FROM Товары CROSS APPLY Склады |

**Пример 1:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Товар\_ID | Масса | Цена | | 1 | 10 | 50 | | 2 | 15 | 100 | | 3 | 5 | 200 | | 4 | 1 | 20 | | 5 | 2 | 250 | | 6 | 50 | 10 | | 7 | 30 | 90 |  |  |  |  | | --- | --- | --- | | Магаз  ин\_ID | Полок | Макс  Масса | | 1 | 3 | 50 | | 2 | 4 | 20 | | CREATE FUNCTION  dbo.f\_Товары\_по\_массе  (@mass as INT, @n INT)  RETURNS TABLE  AS  RETURN  SELECT TOP (@n) WITH TIES Товар\_ID, Цена, Масса  FROM Товары  WHERE Масса <= @mass  ORDER BY Цена asc  GO | SELECT \*  FROM Склады  CROSS APPLY  f\_Товары\_по\_массе(Склады.МаксМасса, Склады.Полок) Товары  ORDER BY Склады.Магазин\_ID, Товары.Цена desc   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Маг..ID | Полок | МаксМасса | Товар\_ID | Цена | Масса | | 1 | 3 | 50 | 1 | 50 | 10 | | 1 | 3 | 50 | 4 | 20 | 1 | | 1 | 3 | 50 | 6 | 10 | 50 | | 2 | 4 | 20 | 3 | 200 | 5 | | 2 | 4 | 20 | 2 | 100 | 15 | | 2 | 4 | 20 | 1 | 50 | 10 | | 2 | 4 | 20 | 4 | 20 | 1 | |

**Пример 2:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Продажи** | | **ндок** | | Дата | | Покупатель\_ID | | **Товар\_ID** | | Колво | | Цена |   SELECT TOP 1 WITH TIES Товар\_ID  FROM Продажи  ORDER BY  Колво\*Цена desc | SELECT distinct Продажи.Покупатель\_ID, Продажи.Товар\_ID  FROM Продажи INNER JOIN  (  SELECT Покупатель\_ID, MAX(Колво\*Цена) МаксСтоим  FROM Продажи  GROUP BY Покупатель\_ID  ) МаксТовар ON  Продажи.Покупатель\_ID = МаксТовар.Покупатель\_ID AND  Продажи.Колво\*Продажи.Цена = МаксСтоим |
| SELECT distinct Продажи.Покупатель\_ID, Продажи.Товар\_ID  FROM Продажи  WHERE Товар\_ID in  (  SELECT TOP 1 WITH TIES Товар\_ID  FROM Продажи T  WHERE T.Покупатель\_ID = **Продажи.Покупатель\_ID**  ORDER BY Колво\*Цена desc  ) |

**Вариант 3:**

SELECT distinct Покупатели.Покупатель\_ID, Товары.Товар\_ID

FROM Покупатели CROSS APPLY

(

SELECT TOP 1 T.Товар\_ID

FROM Продажи T

WHERE T.Покупатель\_ID = **Покупатели.Покупатель\_ID**

ORDER BY Колво\*Цена desc

) AS Товары

Коммент: CROSS APPLY и OUTER APPLY как INNER JOIN и LEFT JOIN.

Что будет, если в Варианте 3 заменить CROSS на OUTER?

**Пример 3:** CROSS APPLY и VALUES

DECLARE @T TABLE (id int, val int)

INSERT INTO @T

VALUES (1, 10), (2, 20), (3, 30), (4, 40)

|  |  |
| --- | --- |
| SELECT \*  FROM @T T  CROSS APPLY (VALUES (1), (2), (3), (4)) as Nums(Num) | SELECT \*  FROM @T T  CROSS APPLY (VALUES(val - 10, val\*5)) as Val(Sub10, Mult5)  WHERE Sub10 > 10 |