

JOBSHEET

**PRAKTIKUM BASIS DATA LANJUT**

**Jurusan Teknologi Informasi POLITEKNIK NEGERI MALANG**

**Week 7**

**SQL SERVER - Window Ranking, Offset, Fungsi Agregat**

1

Information Technology Department, State Polytechnic of Malang **Jobsheet Week -7 : Window Ranking, Offset, Aggregate Function Advanced Database Course (BDL)** Supervisor: Database Teaching Team

# Topics

1. Create a Window with OVER
2. Conducting Function exploration Windows

# Objective

1. Students understand how to explain the T-SQL components used to define windows and the relationship between the two. the
2. Students understand how to write queries using the OVER clause with *partitioning* , *ordering* , and

*framing* to define window

1. Students understand how to write queries using window functions. aggregate
2. Students understand how to write queries using window functions. ranking
3. Students understand how to write queries using window functions. offset

# General Instructions

1. Follow the steps in the practical sections in the order given. given.
2. You can use SQL Server 2012 Standard Edition to try the practicum on this jobsheet. Adjust it to your computer's condition. You.
3. Answer all questions marked [Question-X] found in the specific steps in each section. practicum.
4. In each step of the practicum there is an explanation that will help you answer the questions in instruction number 3, so read and do all the practicum parts in the jobsheet. This.
5. Write the answers to the questions in the instructions number 3 in a report that is done using a word processing application (Word, OpenOffice, or other similar). Export as a **PDF file** with the name format as following:
   * **BDL\_09\_Your\_Full\_Name\_Class** .pdf
   * Example:

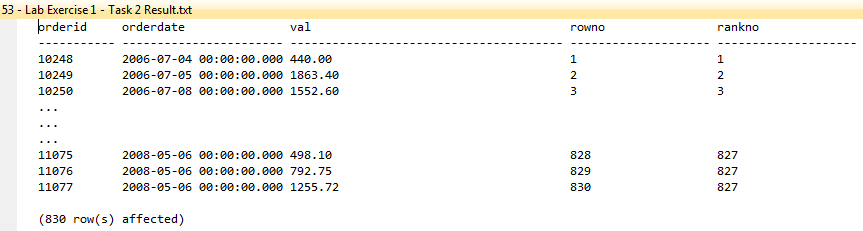
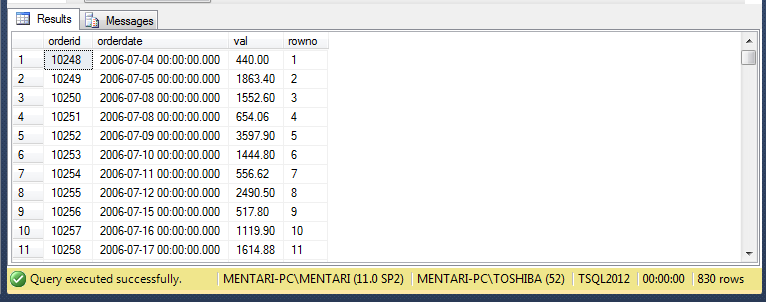
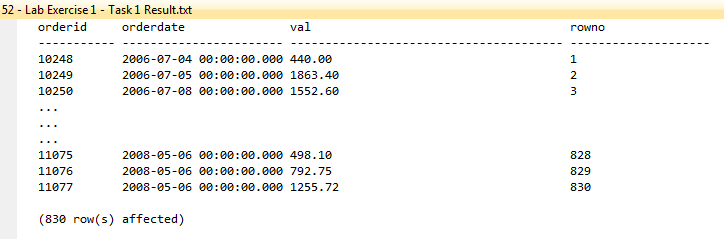
o **BDL\_09\_TI2U\_Mukiyo** .pdf

* + Pay close attention to the format its naming.
  + Collect the PDF files as a practical report to the lecturer. guardian.
  + In addition to the file name, also include your identity on the first page of the report. the.

# Lab – Part 1: Writing Queries Using the RANKING Function

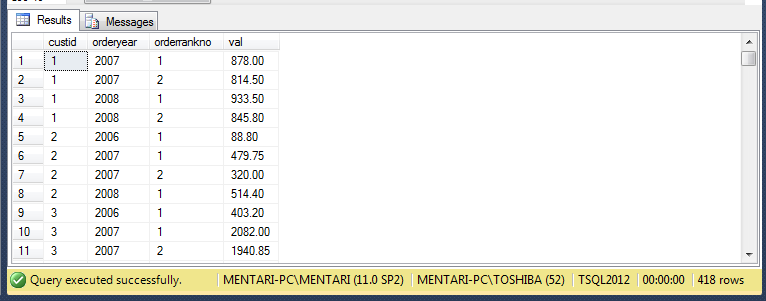
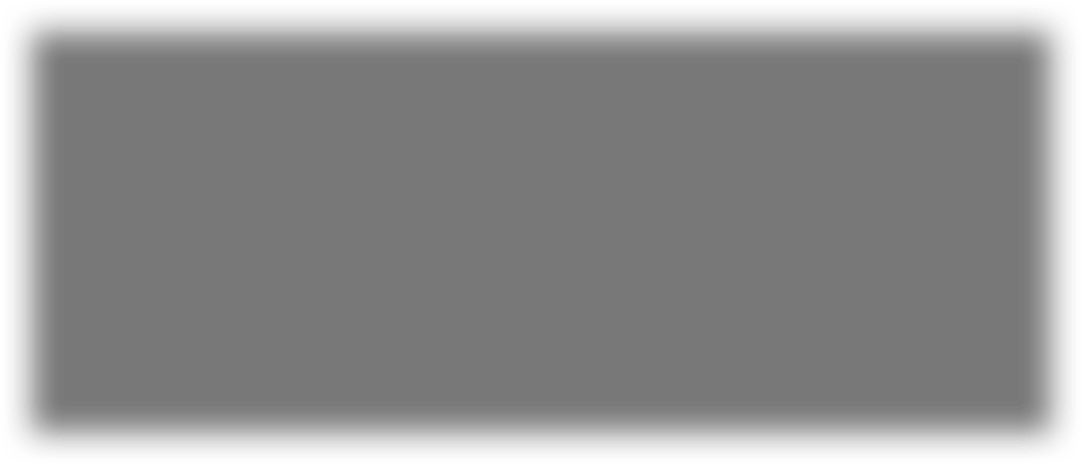
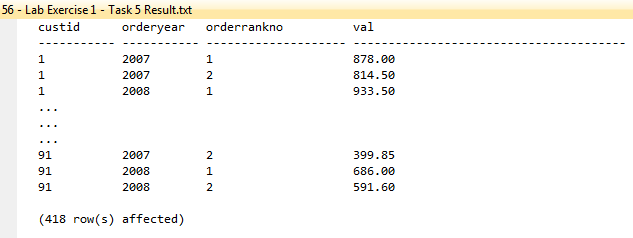
|  |  |
| --- | --- |
| **Step** | **Information** |
| **1** | Scenario :  The sales department wants to determine the order based on the value of each customer. To do this, it is necessary to report using the RANK function (including a calculation result column that adds a calculation result column to display the row number with the SELECT clause).  To do the experiment in this practicum part 1, first log in to SQL Server Management Studio  (SSMS). Make sure the database is connected to “TSQL”. |

|  |  |
| --- | --- |
| **2** | [Question-1] Write a SELECT statement to retrieve the orderid, orderdate, and val columns and a calculated column named rowno from the Sales.OrderValues view! Use the ROW\_NUMBER function to return the rowno, sort the row numbers by the orderdate column! |
| **3** | [Question-2] Copy the T-SQL in question no. 1. Then modify it by inserting an additional column named rankno. To create rankno, use the RANK function with the ranking order based on the orderdate column! |

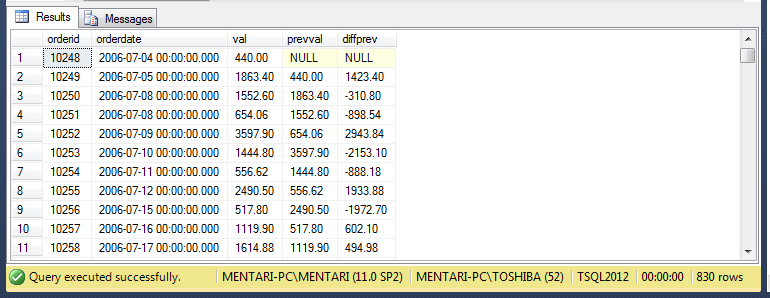
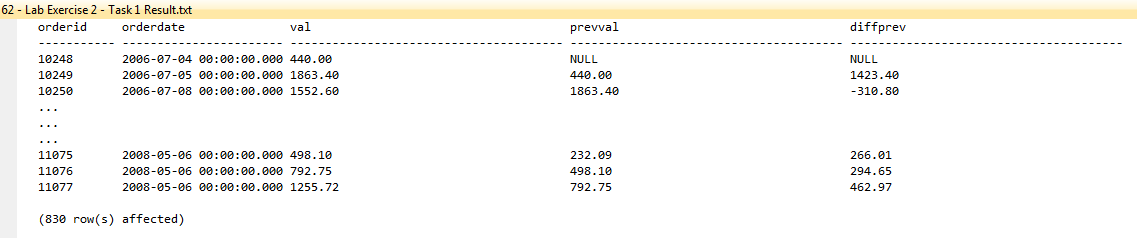


|  |  |
| --- | --- |
|  | * The RANK() function will assign a ranking order based on the orderdate column, and it will give the same rank to rows with identical orderdate values, unlike ROW\_NUMBER(), which assigns unique sequential numbers. |
| **4** | [Question-3] What is the difference between the RANK function and the ROW\_NUMBER function?   *  **ROW\_NUMBER()** does not handle ties and always generates a unique number. *  **RANK()** gives the same rank for ties and skips rank numbers for subsequent rows. |
| **5** | [Question-4] Write a SELECT statement to retrieve the orderid, orderdate, custid, and val columns and calculate a column named orderrankno from the Sales.OrderValues view. The orderrankno column should display the ranking per customer independently, based on the ordering of val in descending order!       * **PARTITION BY custid**: This clause groups the rows by custid, so the ranking is calculated independently for each customer. * **ORDER BY val DESC**: The ranking is ordered by the val column in descending order (higher values get a better rank). * **RANK()**: Assigns the rank within each partition (custid). Tied values will receive the same rank, and gaps will appear in the rank numbering if ties exist. |
| **6** | [Question-5] Write a SELECT statement to retrieve the custid and val columns from the Sales.OrderValues view. Add the following two columns:   1. orderyear as the year of the column order date 2. orderrankno as a sequence number, partitioned by customer and order year, and sorted by order value in descending order. decrease! |

|  |  |
| --- | --- |
|  | * **YEAR(orderdate)**: Extracts the year from the orderdate column and names it orderyear. * **PARTITION BY custid, YEAR(orderdate)**: Partitions the result set by both custid and orderyear, meaning the sequence (rank) will restart for each customer and for each year. * **ORDER BY val DESC**: Orders the rows within each partition by val in descending order. * **ROW\_NUMBER()**: Assigns a sequential number within each partition (customer and year), starting at 1 for each partition. |
| **7** | [Question-6] Copy the query answer to question number 6 And modification to filter only orders with the first two ranks based on column orderrankno! |
| **8** | **Conclusion:** After carrying out this section of the practicum, students know how to use the ranking function in T-SQL statements. |



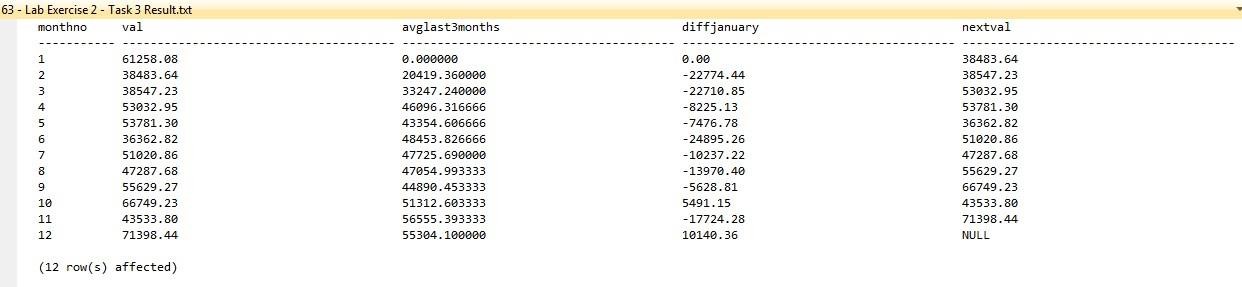
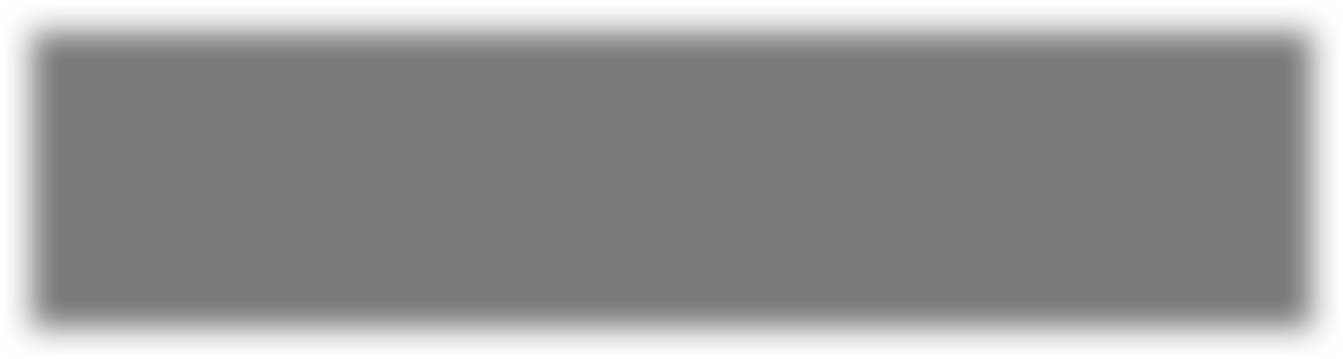
**Lab – Part 2: Writing Queries Using the OFFSET Function**



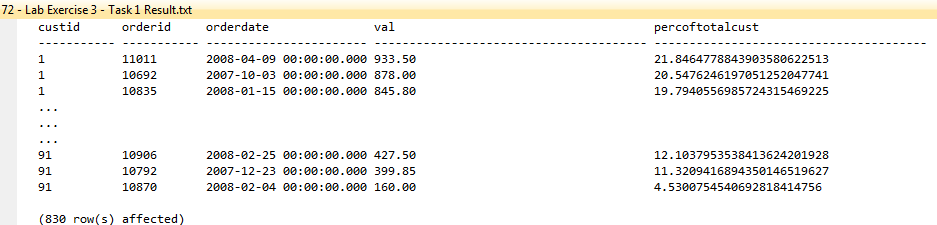
|  |  |
| --- | --- |
| **Step** | **Information** |
| **1** | Scenario :  Another report is needed to analyze the difference between two consecutive rows. This will make it easier for *business users* to analyze growth and trends.  To carry out the experiment in this practical part 2, make sure the database is connected to  “TSQL”. |
| **2** | [Question-7] Create a ( *common table expression* ) CTE named OrderRows based on a query that retrieves the orderid, orderdate, and val columns from the Sales.OrderValues view. Add a calculated result column named rowno using the ROW\_NUMBER function sorted by the orderdate and orderid columns! |
| **3** | [Question-8] Write a SELECT statement against a CTE and use a LEFT JOIN with the same CTE to retrieve the current row *and* previous row *based* on the rowno column. Return the orderid, orderdate, and val columns for the current row and the val column for the previous row as prevval. Add a calculated column named diffprev that shows the difference between the current and previous val! |
| **4** | [Question-9] Write a SELECT statement using the LAG function to get the same results as the query in question no.2! The query created in this problem does not use CTE. |

|  |  |
| --- | --- |
|  |  |
| **5** | [Question-10] Create a CTE named SalesMonth2007 that creates two columns, namely, monthno (the number of months from the orderdate column) and val (the aggregate of the val column)! Then filter the results only for the order year 2007 and group by monthno! |
| **6** | [Question-11] Write a SELECT statement that will take the monthno and val columns from the CTE and add 3 columns to display, namely:   1. avglast3months (average sales amount of three months) final) 2. diffjanuary (difference between current val and val in january, use FIRST\_VALUE function) 3. nextval (value of val column in month furthermore)   Information: The average amount for the last three months is not calculated correctly because the total amount of the first 2 months is divided by 3. |





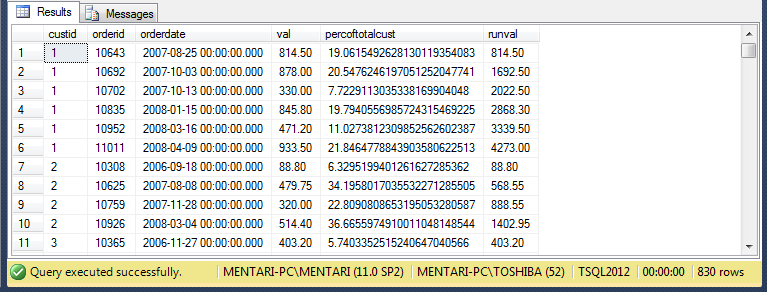
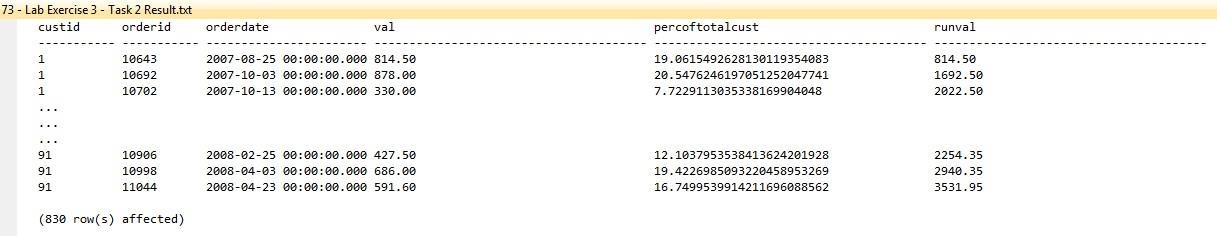
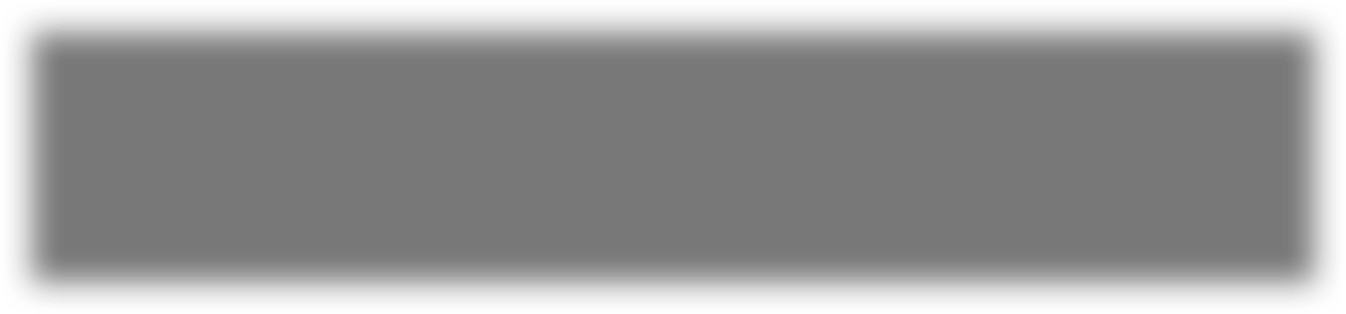
|  |  |
| --- | --- |
|  |  |
| **7** | **Conclusion** : After carrying out this section of the practicum, students can use the OFFSET function in T-SQL statements. |



# Lab – Part 3: Writing Queries Using Window Aggregation Functions

|  |  |
| --- | --- |
| **Step** | **Information** |
| **1** | Scenario :  To better understand the cumulative sales value of customers over time and to provide sales analysts with year-long analysis a different SELECT statement using the window aggregate function is required.  To carry out the experiment in this practical part 3, make sure the database is connected to  “TSQL”. |
| **2** | [Question-12] Write a SELECT statement to retrieve the custid, orderid, orderdate, and val columns from the Sales.OrderValues view. Add a column named percoftotalcust that contains the percentage of each sales order amount compared to the total sales for that customer! |

|  |  |
| --- | --- |
|  |  |
| **3** | [Question-13] Copy the previous SELECT statement and modify it by adding a new calculated column named runval! This column should contain the total sales that have occurred for each customer based on the order date, using orderid as the tiebreaker. |
| **4** | [Question-14] Copy the SalesMonth2007 CTE in experiment 2. Write a SELECT statement to retrieve the monthno and val columns. Add two computed columns:   1. avglast3months. This column should contain the average sales amount for the last three months before the current month using the aggregate window function. Assume that there are no *missing months* . 2. ytdval This column must contain the cumulative sales value up to the current month. This. |



|  |  |
| --- | --- |
|  |  |
| **5** | **Conclusion** : After doing this practical section, you will gain a basic understanding of how to use the window aggregation function in T-SQL statements. |

***--- Have a great time doing it ----***