|  |
| --- |
| Department of Software Engineering  Mehran University of Engineering and Technology, Jamshoro |

|  |  |  |  |
| --- | --- | --- | --- |
| Course: SWE324 - Data Warehousing and Data Mining | | | |
| Instructor | Rabeea Jaffari | **Practical/Lab No.** | 04 |
| Date |  | **CLOs** | CLO-4: P3 & P4 |
| Signature |  | **Assessment Score** | 1 Mark |

|  |  |
| --- | --- |
| Topic | To familiar with OLTP system reporting |
| Objectives | * To learn report generation in OLTP systems |
| Submitted By | * 16SW51 |

**OLTP SYSTEM REPORTS:**  An OLTP systemreport is the formatted result of OLTP system queries and contains useful data for decision-making and analysis and is usually used by managers to get a high-level summary from the transactional data. Most good business applications contain a built-in reporting tool; this is simply a front-end interface that calls or runs back-end OLTP queries that are formatted for easy application usage. However, in case of absence of such tool, one can create his/her own reports by writing all the desired queries, grouping them together in a stored procedure (so that all of them can be executed immediately together) and displaying the returned results in a table. The user can then process the tabular data in other ways (filter/sort, create charts, export to Excel, crosstab, etc.)

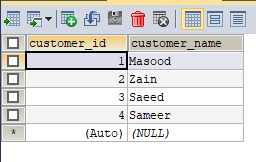
**TOOL:** Although GUI based reporting tools available with most OLTP systems (DBMSs) can be used, this lab focuses on creating customized reports from queries in stored procedures and generating tabular results from them to be displayed in the reports.   
**SQLYog** MYSQL DMBS will be used to achieve the task.

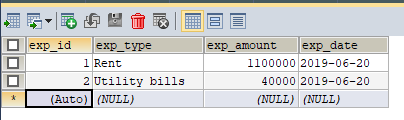
**Task**

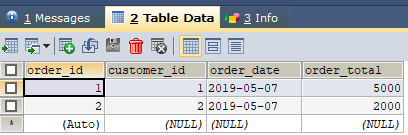
Generate an OLTP system report which displays the following results:

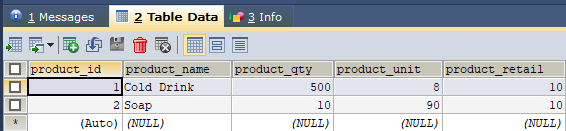
1. **Total sales in a month** (Use only order table).
2. **Profit/loss in a month** (Use product (to account for purchasing costs), expenditure as well as order tables).
3. **Highest selling product of the month** (By highest sold quantity)
4. **Lowest selling product of the month** (By lowest sold quantity)

**Hint:** Create a report table to hold all the above results from queries after they are executed in the stored procedure and then create a stored procedure in the similar manner as shown above.

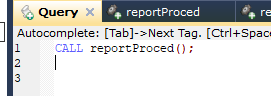


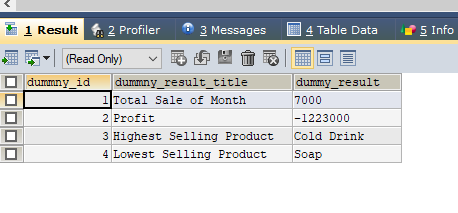












Query

DELIMITER $$

USE `lab4`$$

DROP PROCEDURE IF EXISTS `reportProced`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `reportProced`()

BEGIN

SELECT @total\_sale\_in\_month:= SUM(order\_total) FROM `order` WHERE `order\_date` BETWEEN '2019-05-08' AND '2019-06-22' ;

SELECT @exp:= SUM(`exp\_amount`) FROM `expenditure` WHERE `exp\_date` BETWEEN '2019-05-08' AND '2019-06-22' ;

SELECT @profit:= @total\_sale\_in\_month-@exp;

SELECT @higest\_saling\_product:= p.`product\_name` FROM order\_details o INNER JOIN product p ON p.product\_id=o.product\_id WHERE o.order\_quantity=

(SELECT MAX(order\_quantity) FROM order\_details);

SELECT @lowest\_saling\_product:= p.`product\_name` FROM order\_details o INNER JOIN product p ON p.product\_id=o.product\_id WHERE o.order\_quantity=

(SELECT MIN(order\_quantity) FROM order\_details);

INSERT INTO `dummy\_table` VALUES('','Total Sale of Month',@total\_sale\_in\_month )

,('','Profit',@profit),('','Highest Selling Product',@higest\_saling\_product),

('','Lowest Selling Product',@lowest\_saling\_product);

SELECT \* FROM `dummy\_table`;

END$$

DELIMITER ;