Lab 4

|  |
| --- |
| Lab Tasks |
| Submission Date: 23-04-19 |

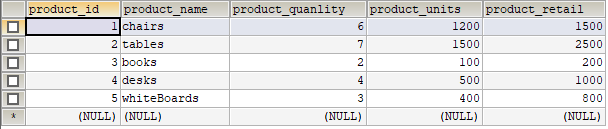
Generate an OLTP system report which displays the following results:

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)

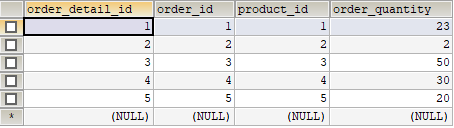
Hint: Create a report table to hold all the above results from queries after they are executed in the stored procedure and then create a stored procedure in the similar manner as shown above.

Solution:

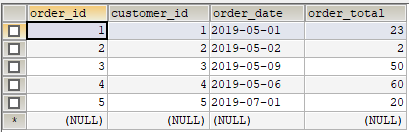
Product table:



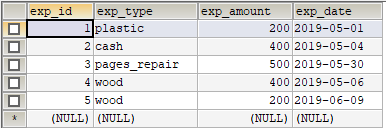
Order\_details table:



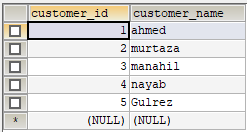
Order table:



Expenditure table:



Customer table



Executed Queries for all above tasks:

1)SELECT SUM(exp\_amount) FROM expenditure WHERE `exp\_date` LIKE '2019-05-04' AND '2019-05-09';

2)SELECT SUM(order\_total) AS totalSale FROM `order` WHERE `order\_date` LIKE '2019-07%';

3)SELECT (SUM(product.product\_retail)-(SUM(product.product\_units)+SUM(expenditure.exp\_amount))) AS Profit\_Loss FROM

product product INNER JOIN expenditure expenditure ON product.product\_id= expenditure.exp\_id WHERE expenditure.exp\_date IN

(SELECT order.order\_date FROM `order` `order` HAVING order.order\_date BETWEEN '2019-05-04' AND '2019-05-09');

4)SELECT product.product\_name FROM product product INNER JOIN order\_details order\_details ON

product.product\_id=(SELECT order\_details.product\_id WHERE order\_details.order\_quantity=

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

5)SELECT product.product\_name FROM product product INNER JOIN order\_details order\_details ON

product.product\_id=(SELECT order\_details.product\_id WHERE order\_details.order\_quantity=

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

INSERT INTO result(`type`) VALUES (lowest\_sale\_product'),('highest\_sale\_product');

Report of all above task:

DELIMITER $$

CREATE

/\*[DEFINER = { user | CURRENT\_USER }]\*/

PROCEDURE `reportdb`.`report`()

/\*LANGUAGE SQL

| [NOT] DETERMINISTIC

| { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }

| SQL SECURITY { DEFINER | INVOKER }

| COMMENT 'string'\*/

BEGIN

SELECT @total\_expend:=SUM(exp\_amount) AS total\_expend FROM expenditure WHERE `exp\_date` LIKE '2019-05-01' AND '2019-05-30';

INSERT INTO result(total\_expend) VALUES (@total\_expend);

SELECT @total\_sale:=SUM(order\_total) AS total\_sale FROM `order` WHERE `order\_date` LIKE '2019-05%';

INSERT INTO result(total\_sale) VALUES (@total\_sale);

SELECT @profit\_loss:=(SUM(product.product\_retail)-(SUM(product.product\_units)+SUM(expenditure.exp\_amount))) AS Profit\_Loss FROM

product product INNER JOIN expenditure expenditure ON product.product\_id= expenditure.exp\_id WHERE expenditure.exp\_date IN

(SELECT order.order\_date FROM `order` `order` HAVING order.order\_date BETWEEN '2019-05-01' AND '2019-05-30');

INSERT INTO result(profit\_loss) VALUES (@profit\_loss);

SELECT \* FROM result;

SELECT @low\_sale\_product:=product.product\_name FROM product product INNER JOIN order\_details order\_details ON

product.product\_id=(SELECT order\_details.product\_id WHERE order\_details.order\_quantity=(SELECT MIN(order\_details.order\_quantity)

FROM order\_details order\_details));

INSERT INTO result(lowest\_sale\_product) VALUES (@low\_sale\_product);

SELECT \* FROM result;

SELECT @high\_sale\_product:=product.product\_name FROM product product INNER JOIN order\_details order\_details ON

product.product\_id=(SELECT order\_details.product\_id WHERE order\_details.order\_quantity=

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

INSERT INTO result(high\_sale\_product) VALUES (@high\_sale\_product);

SELECT \* FROM result;

END$$

DELIMITER ;

