<u>מימוש הפעולות ב sql</u>

- מסמך הכולל: לכל פעולה את הקוד ב SQL -שמממש אותה.

1. Book X exist in stock?

pstmt = con->prepareStatement("SELECT * FROM stock WHERE BookID=?");

```
    pstmt->setInt(1, bookId);

    pstmt->executeUpdate();

    res = pstmt->executeQuery();
```

2. Who is earliest customer?

res = stmt->executeQuery("SELECT * FROM customers ORDER BY CreatedAt ASC LIMIT 1");

3. What is the oldest book?

res = stmt->executeQuery("SELECT * FROM stock ORDER BY EnteredOn ASC LIMIT 1");

```
    if (res->next()) {

        cout << "The earliest book in stock is " << res->getInt("BookID")
<< endl;

    }
```

else {

cout << "There are no books in stock" << endl;

4. Print order list

res = stmt->executeQuery("SELECT * FROM orders ORDER BY CreatedAt DESC");

printTable(OrdersFieldsList, res);

5 .How many copies of book Y have been sold?

int bookId;

cout << "Enter Book ID" << endl;

cin >> bookId;

pstmt = con->prepareStatement("SELECT COUNT(*) FROM sales WHERE BookID=?");

pstmt->setInt(1, bookId);

pstmt->executeUpdate();

res = pstmt->executeQuery();

6. What is the common author between dates X to Y?

string query = " \

SELECT * \\ FROM (\\SELECT a.AuthorID, Name, COUNT(*) as counter \

FROM sales s JOIN booksAuthors ba ON s.BookID = ba.BookID JOIN authors a ON ba.AuthorID = a.AuthorID \

WHERE CreatedAt BETWEEN '" + from + "' AND '" + to + "' GROUP BY a.AuthorID, Name \

**) t **

ORDER BY t.counter DESC \

LIMIT 1";

7. Who are the 3 customers that bought the largest amount of book ever?

SELECT * \\FROM customers c JOIN (\\SELECT s.CustomerID \

FROM sales s JOIN customers c ON s.CustomerID = c.CustomerID GROUP BY s.CustomerID ORDER BY COUNT(*) DESC LIMIT 3 \

) t ON c.CustomerID = t.CustomerID \

8. Who is the book with the largest amount of translations in stock?

string query = " \

SELECT b.Title, COUNT(distinct b.Translator) as num FROM stock s JOIN books b ON s.BookID = b.BookID \

GROUP BY b.BookGroup, b.Title \

ORDER BY num DESC \\LIMIT 1 \

9. Buying history of customer X

string query = " \

SELECT * \

FROM sales \

WHERE CustomerID = " + to_string(customerId) + " ORDER BY CreatedAt DESC";

10. Show order list of customer X

string query = "SELECT * FROM orders WHERE CustomerID=" + to_string(customerId) + " ORDER BY CreatedAt DESC";

11 .Calculate price of delivery

string query = "SELECT o.*, b.Weight FROM orderTmp o JOIN books b ON o.BookID = b.BookID";

12. Does customer X split its purchase to different deliveries?

**string query = " \SELECT s.* FROM sales s JOIN \(SELECT s.SaleID FROM sales s JOIN salesDeliveries sd ON s.SaleDeliveryID = sd.SaleDeliveryID WHERE s.CustomerID = " + to_string(customerId) + " GROUP BY s.SaleID HAVING COUNT(distinct sd.SaleDeliveryID) > 1) t **

**ON s.SaleID = t.SaleID **

**ORDER BY s.SaleID **

13 .Show current status of delivery

string query = "SELECT * FROM salesDeliveries WHERE SaleDeliveryID = '" + code + "'";

14 .What is the sum of all the deliveries of Xpress in month X?

string query = "SELECT CONVERT(ROUND(SUM(s.Payment),2),CHAR) as Total FROM sales s JOIN salesDeliveries sd ON s.SaleDeliveryID = sd.SaleDeliveryID WHERE sd.DeliveryType LIKE 'XPRESS%' AND MONTH(s.CreatedAt) = " + month + " AND YEAR(s.CreatedAt) = " + year;

15. What is the sum of all the Bit transfers to the shop in month Y?

string query = "SELECT CONVERT(ROUND(SUM(Payment),2),CHAR) as Total FROM sales WHERE PaymentType = 'BIT' AND MONTH(CreatedAt) = " + month + " AND YEAR(CreatedAt) = " + year;

res = stmt->executeQuery(query.c_str());

res->next();

cout << "The sum of all the BIT payments to the store is " << res->getString("Total").c_str() << endl;

16. List all deals in the last 12 months, that their profit is bigger than the average

string query = "SELECT ROUND(SUM(Payment)/COUNT(*),2) as res FROM sales WHERE CreatedAt >= DATE_SUB(NOW(), INTERVAL 12 MONTH)";

res = stmt->executeQuery(query.c_str());

res->next();

double avg = res->getDouble("res");

query = "SELECT * FROM sales WHERE CreatedAt >= DATE_SUB(NOW(), INTERVAL 12 MONTH) AND Payment > " + to_string(avg);

res = stmt->executeQuery(query.c_str());

printTable(SalesFieldsList, res);

17. How many deliveries performed by Israel Post and Xpress in the last 12 months?

string query = "SELECT COUNT(distinct s.SaleDeliveryID) as Total FROM sales s JOIN salesDeliveries sd ON s.SaleDeliveryID = sd.SaleDeliveryID

WHERE sd.DeliveryType LIKE 'XPRESS%' AND s.CreatedAt >= DATE_SUB(NOW(), INTERVAL 12 MONTH)";

res = stmt->executeQuery(query.c_str());

res->next();

cout << "The num of Xpress deliveries is " << res->getString("Total").c_str() << endl;

query = "SELECT COUNT(distinct s.SaleDeliveryID) as Total FROM sales s JOIN salesDeliveries sd ON s.SaleDeliveryID = sd.SaleDeliveryID WHERE sd.DeliveryType LIKE 'IPO%' AND s.CreatedAt >= DATE_SUB(NOW(), INTERVAL 12 MONTH)";;

res = stmt->executeQuery(query.c_str());

res->next();

cout << "The num of Israel Post Office deliveries is " << res->getString("Total").c_str() << endl;

delete res;

18. List all the deliveries that include at least 2 editions of the same book

string query = " \

SELECt s.* FROM sales s JOIN \

(SELECT s.SaleID, b.BookGroup, COUNT(*) as num FROM sales s JOIN books b ON s.BookID = b.BookID GROUP BY s.SaleID, b.BookGroup HAVING COUNT(*) > 1) t \

ON s.SaleID = t.SaleID \

";

19. List all customers that bought a book once, but didn't purchase in the last 24 months

string query = " \

SELECT * FROM customers c JOIN \

(SELECT distinct s.CustomerID FROM sales s WHERE s.CustomerID NOT IN(SELECT distinct CustomerID FROM sales ss WHERE ss.CreatedAt >= DATE_SUB(NOW(), INTERVAL 24 MONTH))) t \

ON c.CustomerID = t.CustomerID \

";

20. List all customers that the shop told them that the book is available 14 days ago but they still didn't purchase the book

string query = " \

SELECT c.* FROM customers c JOIN \

(SELECT DISTINCT CustomerID FROM orders WHERE Status = 'CONTACT' AND ContactAt < DATE_SUB(NOW(), INTERVAL 14 DAY)) t ON c.CustomerID = t.CustomerID \

";

21. Number of books in the stock in each month

string query = "SELECT COUNT(*) as num FROM stock WHERE MONTH(EnteredOn) = " + to_string(i) + " AND Location = 'STOCK'";

22. How many books the store purchased between D1 to D2 and what is the total amount of payment?

string query = "SELECT COUNT(*) as counter, SUM(Price) as total FROM shopBuy WHERE BuyAt BETWEEN '" + from + "' AND '" + to + "'";

cout << query << endl;

res = stmt->executeQuery(query.c_str());

23. What is the profit in year Y and month X?

string query = "SELECT ROUND(SUM(Payment),2) as TotalSales FROM sales WHERE MONTH(CreatedAt) = " + month + " AND YEAR(CreatedAt) = " + year;

```
        res = stmt->executeQuery(query.c_str());

        res->next();

        double totalSales = res->getDouble("TotalSales");
```

query = "SELECT ROUND(SUM(Price),2) as TotalBuy FROM shopBuy WHERE MONTH(BuyAt) = " + month + " AND YEAR(BuyAt) = " + year;

```
        res = stmt->executeQuery(query.c_str());

        res->next();

        double totalBuy = res->getDouble("TotalBuy");


        cout << "The total profit is " << (totalSales - totalBuy) << endl;

}
```

24. Show average of deals in every month

string query = "SELECT ROUND((SUM(Payment)/COUNT(*)),2) as res FROM sales WHERE MONTH(CreatedAt) = " + to_string(i);

```
                res = stmt->executeQuery(query.c_str());
```

25. Show salary of worker Z in month X and year Y

```
string query = "SELECT NumHours, MoneyPerHour FROM salesmenHours WHERE SalesmenID = " + salesmanId + " AND Year =" + year + " AND Month = " + month;


res = stmt->executeQuery(query.c_str());

res->next();

cout << "The salary is " << res->getInt("NumHours") * res->getInt("MoneyPerHour") << endl;
```

26. Who is the top seller in month X and year Y?

```
string query = " \

SELECT SalesmenID, COUNT(*) as num \

FROM sales WHERE MONTH(CreatedAt) = 1 AND YEAR(CreatedAt) = 2020 \

GROUP BY SalesmenID \

ORDER BY COUNT(*) DESC LIMIT 1 \

";

res = stmt->executeQuery(query.c_str());

res->next();

cout << "The top seller is " << res->getInt("SalesmenID") << " with " << res->getInt("num") << " sales" << endl;
```