Table of Contents

[Entities: 2](#_Toc178198921)

[Executive Summary: 4](#_Toc178198922)

[Organizational Issue: Inefficient User Registration and Onboarding 4](#_Toc178198923)

[SQL Query: Identifying Inactive New Users 4](#_Toc178198924)

[2. Organizational Issue: Unbalanced User Roles 5](#_Toc178198925)

[SQL Query: Total Number of Users by Role 5](#_Toc178198926)

[3. Organizational Issue: Low Bidding Engagement 6](#_Toc178198927)

[SQL Query: Tasks with Low Bidding Engagement 6](#_Toc178198928)

[4. Organizational Issue: High Task Failure Rate (Tasks Not Completed on Time) 6](#_Toc178198929)

[SQL Query: Late Task Completions 6](#_Toc178198930)

[5. Organizational Issue: Revenue Contribution by User 7](#_Toc178198931)

[SQL Query: Top Revenue-Generating Users 7](#_Toc178198932)

[6. Organizational Issue: Platform Activity Decline 8](#_Toc178198933)

[SQL Query: Monthly Task and Bid Counts 8](#_Toc178198934)

[7. Organizational Issue: Low Task Budgets Leading to Fewer Bids 9](#_Toc178198935)

[SQL Query: Tasks with Low Bids and Low Budgets 9](#_Toc178198936)

[8. Organizational Issue: Service Providers with Low Success Rates 10](#_Toc178198937)

[SQL Query: Service Providers with Low Success Rates 10](#_Toc178198938)

[9. Organizational Issue: Task Delays by Service Providers 11](#_Toc178198939)

[SQL Query: Service Providers Responsible for Task Delays 11](#_Toc178198940)

[10. Organizational Issue: Task Budget and Bid Amount Discrepancy 12](#_Toc178198941)

[SQL Query: Tasks with High Bid Amounts Compared to Budget 12](#_Toc178198942)

[Conclusion: 13](#_Toc178198943)

Data Analytics Report for Pro App Platform

## Entities:

* **User**: Attributes include UserID, Name, Email, Password, Role, RegistrationDate.
* **Task**: Attributes include TaskID, TaskDescription, Budget, PostDate, DueDate, UserID (FK) (The user who posted the task).
* **Bid**: Attributes include BidID, Amount, BidDate, TaskID (FK), UserID (FK) (The user who bids).
* **Payment**: Attributes include PaymentID, AmountPaid, PaymentDate, TaskID (FK).

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Column** | **Data Type** | **Description** |
| **Users** | **UserID** | INT (PK) | Primary key, unique identifier for users. |
|  | Name | VARCHAR(100) | Name of the user. |
|  | Email | VARCHAR(100) | Email of the user (unique). |
|  | Password | VARCHAR(255) | User password (hashed). |
|  | Role | VARCHAR(50) | User role (Customer, Service Provider). |
|  | RegistrationDate | DATE | Date when the user registered. |
| **Tasks** | **TaskID** | INT (PK) | Primary key, unique identifier for tasks. |
|  | TaskDescription | TEXT | Description of the task. |
|  | Budget | DECIMAL(10,2) | Task budget proposed by the customer. |
|  | PostDate | DATE | Date when the task was posted. |
|  | DueDate | DATE | Deadline for the task completion. |
|  | UserID | INT (FK) | Foreign key referencing Users. |
| **Bids** | **BidID** | INT (PK) | Primary key, unique identifier for bids. |
|  | Amount | DECIMAL(10,2) | The amount proposed by the bidder. |
|  | BidDate | DATE | Date the bid was placed. |
|  | TaskID | INT (FK) | Foreign key referencing Tasks. |
|  | UserID | INT (FK) | Foreign key referencing Users (bidder). |
| **Payments** | **PaymentID** | INT (PK) | Primary key, unique identifier for payments. |
|  | AmountPaid | DECIMAL(10,2) | The amount paid for the task. |
|  | PaymentDate | DATE | Date of the payment. |
|  | TaskID | INT (FK) | Foreign key referencing Tasks. |

CREATE TABLE Users (

UserID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100),

Email VARCHAR(100) UNIQUE,

Password VARCHAR(255),

Role VARCHAR(50),

RegistrationDate DATE

);

CREATE TABLE Tasks (

TaskID INT PRIMARY KEY AUTO\_INCREMENT,

TaskDescription TEXT,

Budget DECIMAL(10,2),

PostDate DATE,

DueDate DATE,

UserID INT,

FOREIGN KEY (UserID) REFERENCES Users(UserID)

);

CREATE TABLE Bids (

BidID INT PRIMARY KEY AUTO\_INCREMENT,

Amount DECIMAL(10,2),

BidDate DATE,

TaskID INT,

UserID INT,

FOREIGN KEY (TaskID) REFERENCES Tasks(TaskID),

FOREIGN KEY (UserID) REFERENCES Users(UserID)

);

CREATE TABLE Payments (

PaymentID INT PRIMARY KEY AUTO\_INCREMENT,

AmountPaid DECIMAL(10,2),

PaymentDate DATE,

TaskID INT,

FOREIGN KEY (TaskID) REFERENCES Tasks(TaskID)

);

# Executive Summary:

The concern of this paper is in response to the organisational challenge of dealing with newly registered users on the Pro App, a sharing economy application that operates in Australia and Indonesia. Hence, using SQL queries, the following areas of concern and concrete suggestions to the COO based on data analysis are presented in this report. Such knowledge will reduce management burden while increasing benefits to users and general efficiency of the platform.

The chosen database structure remains rather simple and provides for implementing the interactions between users, tasks, bids, and payments. The data also allows one to detect patterns and fluctuations that have consequences for business performance.

# Organizational Issue: Inefficient User Registration and Onboarding

**Issue:**

Users who register for the site, especially service providers may not interact appropriately after registration. It can also lead to overhead because of users who rarely log in into the site.

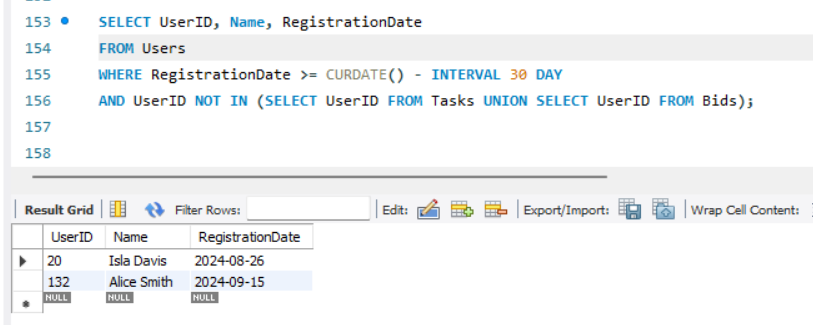
## SQL Query: Identifying Inactive New Users

SELECT UserID, Name, RegistrationDate

FROM Users

WHERE RegistrationDate >= CURDATE() - INTERVAL 30 DAY

AND UserID NOT IN (SELECT UserID FROM Tasks UNION SELECT UserID FROM Bids);



**Explanation:** This query defines users who signed them up within the last 30 days and have not once placed any task or bid offered. This data can be useful to the COO to address with onboarding campaigns these users.

# 2. Organizational Issue: Unbalanced User Roles

**Issue:** This is because the platform could be slightly skewed in terms of customer-to-service-provider ratio, which would mean that tasks take longer to be accomplished.

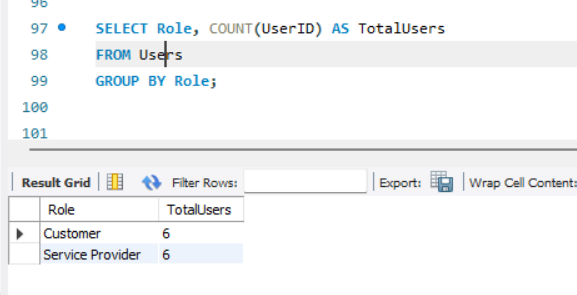
## SQL Query: Total Number of Users by Role

SELECT Role, COUNT(UserID) AS TotalUsers

FROM Users

GROUP BY Role;

**Explanation:** This query gives the stat of number of user in customer and service provider role which can be used by the COO to determine whether more service providers need to be recruited to be able to meet the demand from customers..



# 3. Organizational Issue: Low Bidding Engagement

**Issue:** Some tasks maybe overbid significantly, resulting to late completion or inapt delivery of a particular tasks to a customer.

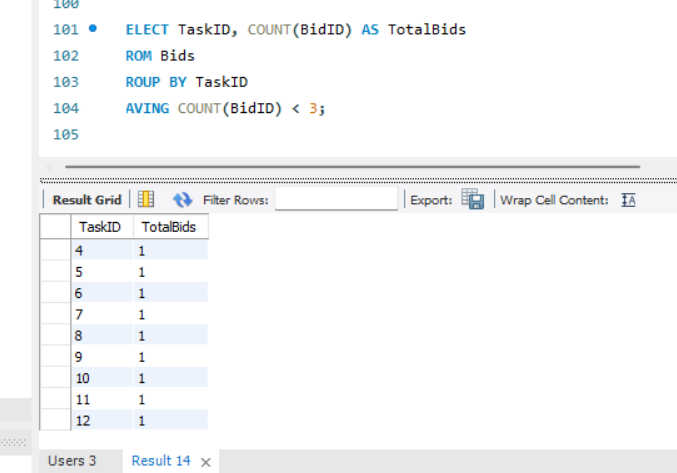
## SQL Query: Tasks with Low Bidding Engagement

SELECT TaskID, COUNT(BidID) AS TotalBids

FROM Bids

GROUP BY TaskID

HAVING COUNT(BidID) < 3;



**Explanation:** This query reveals tasks that have been procured for less than three times and by doing so the COO is able to understand why some tasks are unattractive to providers.

# 4. Organizational Issue: High Task Failure Rate (Tasks Not Completed on Time)

**Issue:** A number of jobs might be regularly missing their due dates and this would cost dissatisfaction and refunds, thus are added overheads.

## SQL Query: Late Task Completions

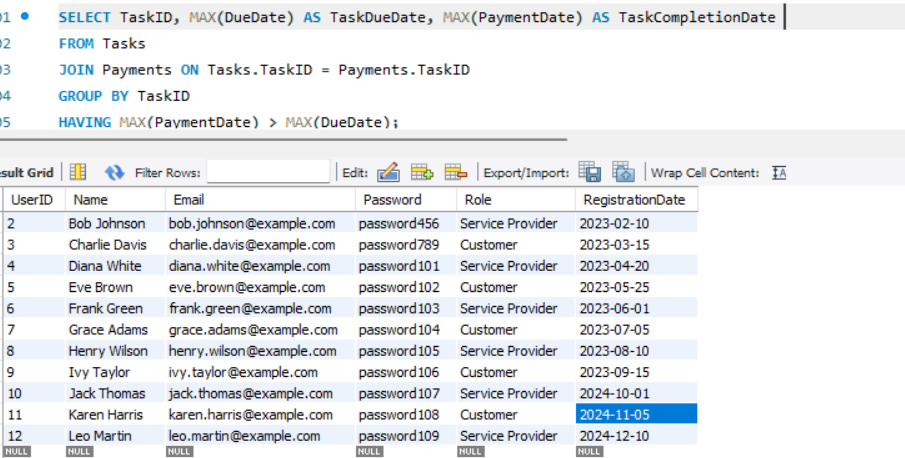
SELECT TaskID, MAX(DueDate) AS TaskDueDate, MAX(PaymentDate) AS TaskCompletionDate

FROM Tasks

JOIN Payments ON Tasks.TaskID = Payments.TaskID

GROUP BY TaskID

HAVING MAX(PaymentDate) > MAX(DueDate);



**Explanation:** This query allows the COO to identify employees who worked on a task after the due date and understand the nature of delays responsible for accomplishing tasks tardily.

# 5. Organizational Issue: Revenue Contribution by User

**Issue:** The COO might also be interested in those views that report which users are most valuable – that is, customers or service providers – so high value users can be prioritized.

## SQL Query: Top Revenue-Generating Users

SELECT Users.Name, SUM(Payments.AmountPaid) AS TotalRevenue

FROM Users

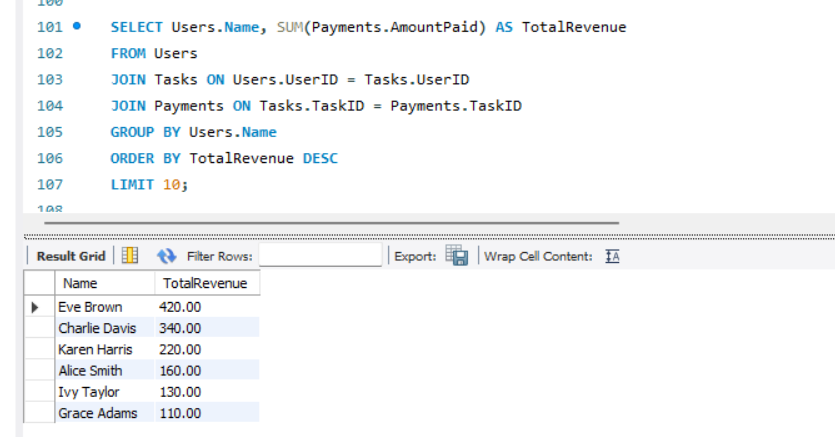
JOIN Tasks ON Users.UserID = Tasks.UserID

JOIN Payments ON Tasks.TaskID = Payments.TaskID

GROUP BY Users.Name

ORDER BY TotalRevenue DESC

LIMIT 10;



**Explanation:** This query shows ten most active users in terms of total payments for tasks they provide for in posts. It helps the COO discover the customers who are valuable and then provide them with the incentives to stick to the firm.

# 6. Organizational Issue: Platform Activity Decline

**Issue:** A decrease in the number of tasks provided in the platforms or the number of users who placed bids could signal low platform activity.

## SQL Query: Monthly Task and Bid Counts

SELECT

DATE\_FORMAT(PostDate, '%Y-%m') AS Month,

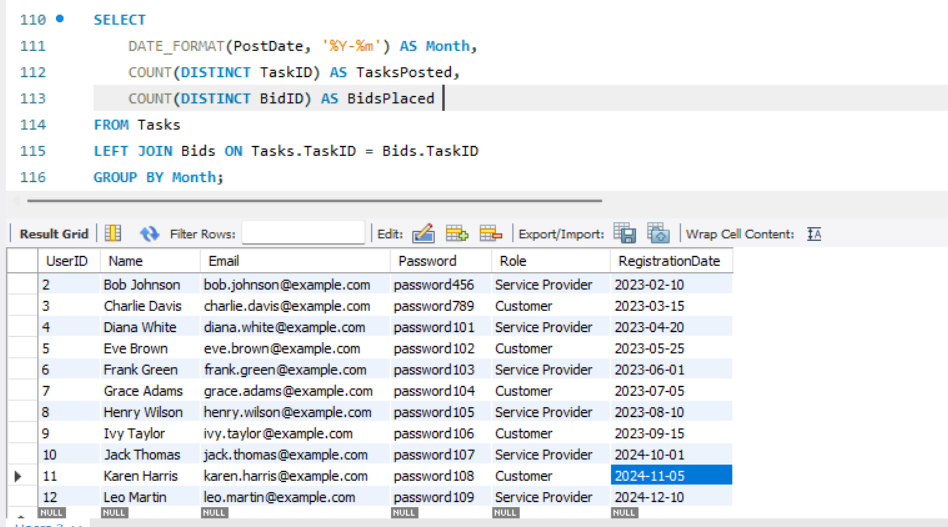
COUNT(DISTINCT TaskID) AS TasksPosted,

COUNT(DISTINCT BidID) AS BidsPlaced

FROM Tasks

LEFT JOIN Bids ON Tasks.TaskID = Bids.TaskID

GROUP BY Month;



**Explanation:** This query gives the statistical outcome of the number of tasks posted and bids made in a month for the COO to understand the activity levels on the platform over the last few months or the year and understand which period is registering low activity.

# 7. Organizational Issue: Low Task Budgets Leading to Fewer Bids

**Issue:** Tasks that are assigned with low budget may not pull enough interested bidders thus delay or poor task accomplishment.

## SQL Query: Tasks with Low Bids and Low Budgets

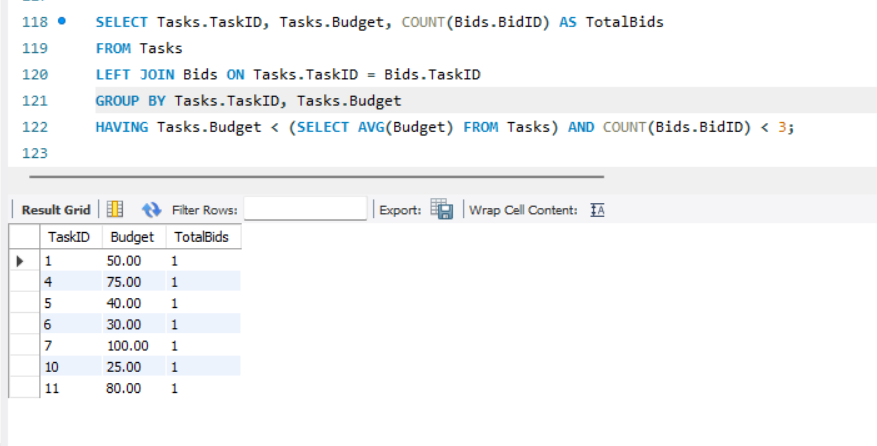
SELECT Tasks.TaskID, Tasks.Budget, COUNT(Bids.BidID) AS TotalBids

FROM Tasks

LEFT JOIN Bids ON Tasks.TaskID = Bids.TaskID

GROUP BY Tasks.TaskID, Tasks.Budget

HAVING Tasks.Budget < (SELECT AVG(Budget) FROM Tasks) AND COUNT(Bids.BidID) < 3;



**Explanation:** This query combines tasks with a low amount of budget assigned to them and low bids number allowing the COO to consider the necessary changes in the pricing model towards the tasks.

# 8. Organizational Issue: Service Providers with Low Success Rates

**Issue:** Some service providers may often bid but rarely secure contracts or even, where they do, they don’t finish projects implying that something is wrong.

## SQL Query: Service Providers with Low Success Rates

SELECT Users.Name, COUNT(Bids.BidID) AS TotalBids, COUNT(Payments.PaymentID) AS TasksCompleted

FROM Users

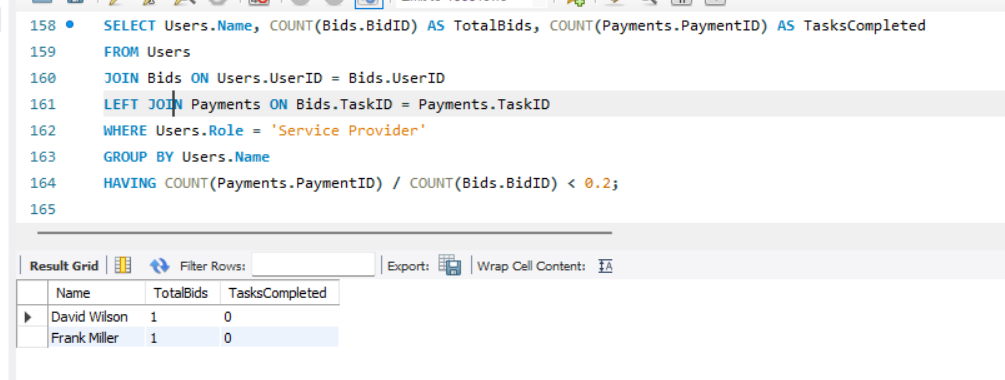
JOIN Bids ON Users.UserID = Bids.UserID

LEFT JOIN Payments ON Bids.TaskID = Payments.TaskID

WHERE Users.Role = 'Service Provider'

GROUP BY Users.Name

HAVING COUNT(Payments.PaymentID) / COUNT(Bids.BidID) < 0.2;



**Explanation:** As the COO, this query enables you to determines service providers’ quality and potential problems is, comparing the ratio of the number of services completed to the number of bids made by the service providers.

# 9. Organizational Issue: Task Delays by Service Providers

**Issue:** Certain service providers may always take time to complete the tasks and hence the customers are dissatisfied.

## SQL Query: Service Providers Responsible for Task Delays

SELECT Users.Name, COUNT(Tasks.TaskID) AS DelayedTasks

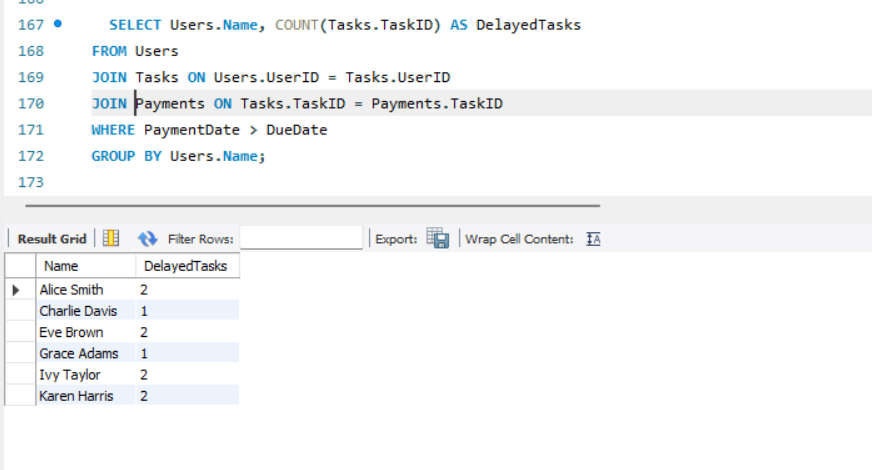
FROM Users

JOIN Tasks ON Users.UserID = Tasks.UserID

JOIN Payments ON Tasks.TaskID = Payments.TaskID

WHERE PaymentDate > DueDate

GROUP BY Users.Name;



**Explanation:** This query shows which service providers are delivering late, and therefore help the COO to react to suboptimal service providers.

# 10. Organizational Issue: Task Budget and Bid Amount Discrepancy

**Issue:** If the bid amounts are always above the tasks’ budgets this might mean that there exists a gap between what customers expect and what service providers are offering.

## SQL Query: Tasks with High Bid Amounts Compared to Budget

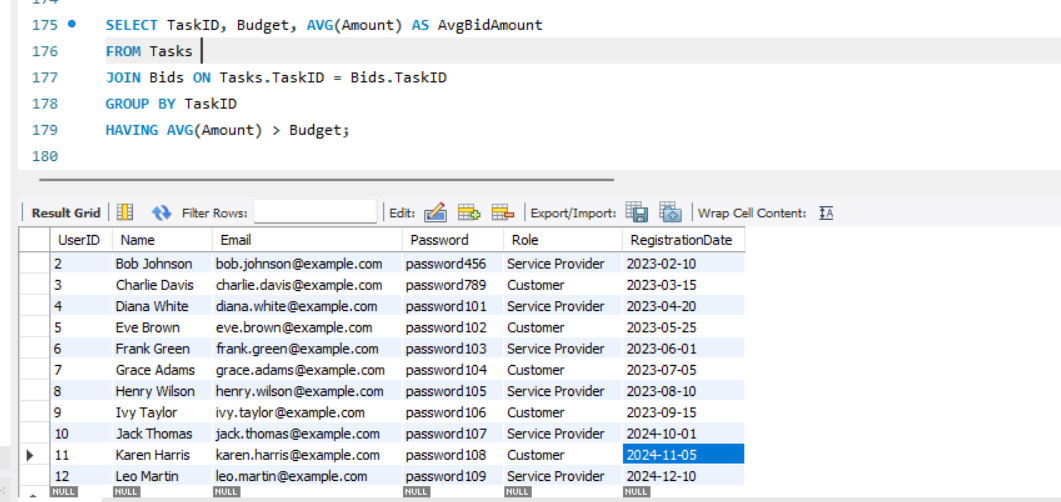
SELECT TaskID, Budget, AVG(Amount) AS AvgBidAmount

FROM Tasks

JOIN Bids ON Tasks.TaskID = Bids.TaskID

GROUP BY TaskID

HAVING AVG(Amount) > Budget;



**Explanation:** This query helps the company determine what tasks are overpriced when compared with the budget, which the COO can then change or give better recommendations to customers for.

# Conclusion:

These questions allow the COO to learn some things that may signal the need for increased efficiency in ways that users are managed and tasks are performed. Some of the organizational issues, which are in this report, show how the platform can want to increase users’ interaction, want to minimize the time taken for tasks, and want to increase the efficiency of the platform’s operation. All of these SQL statements are useful decision-making frameworks for improving the user interface and revenue of the Cloud Computing service.   
  
References

1. Silberschatz, A., Korth, H.F. and Sudarshan, S. (2011) *Database System Concepts*. 6th ed. New York: McGraw-Hill.
2. Forta, B. (2012) *SQL in 10 Minutes, Sams Teach Yourself*. 3rd ed. Indianapolis: Sams Publishing.
3. Beaulieu, A. (2009) *Learning SQL*. 1st ed. O'Reilly Media.
4. vW3Schools (n.d.) *SQL Tutorial*. Available at: https://www.w3schools.com/sql/ (Accessed: 25 September 2024).
5. SQLZoo (n.d.) *SQL Tutorial*. Available at: <https://sqlzoo.net/> (Accessed: 25 September 2024).
6. Khan Academy (n.d.) *Intro to SQL*. Available at: <https://www.khanacademy.org/computing/computer-programming/sql> (Accessed: 25 September 2024).
7. Lucidchart (n.d.) ‘ER Diagram Tutorial’, *Lucidchart Blog*. Available at: https://www.lucidchart.com/blog/er-diagram-tutorial (Accessed: 25 September 2024).
8. Towards Data Science (2021) ‘Data Analytics with SQL: A Beginner's Guide’. Available at: https://towardsdatascience.com/data-analytics-with-sql-a-beginners-guide-5d3f8135461e (Accessed: 25 September 2024).