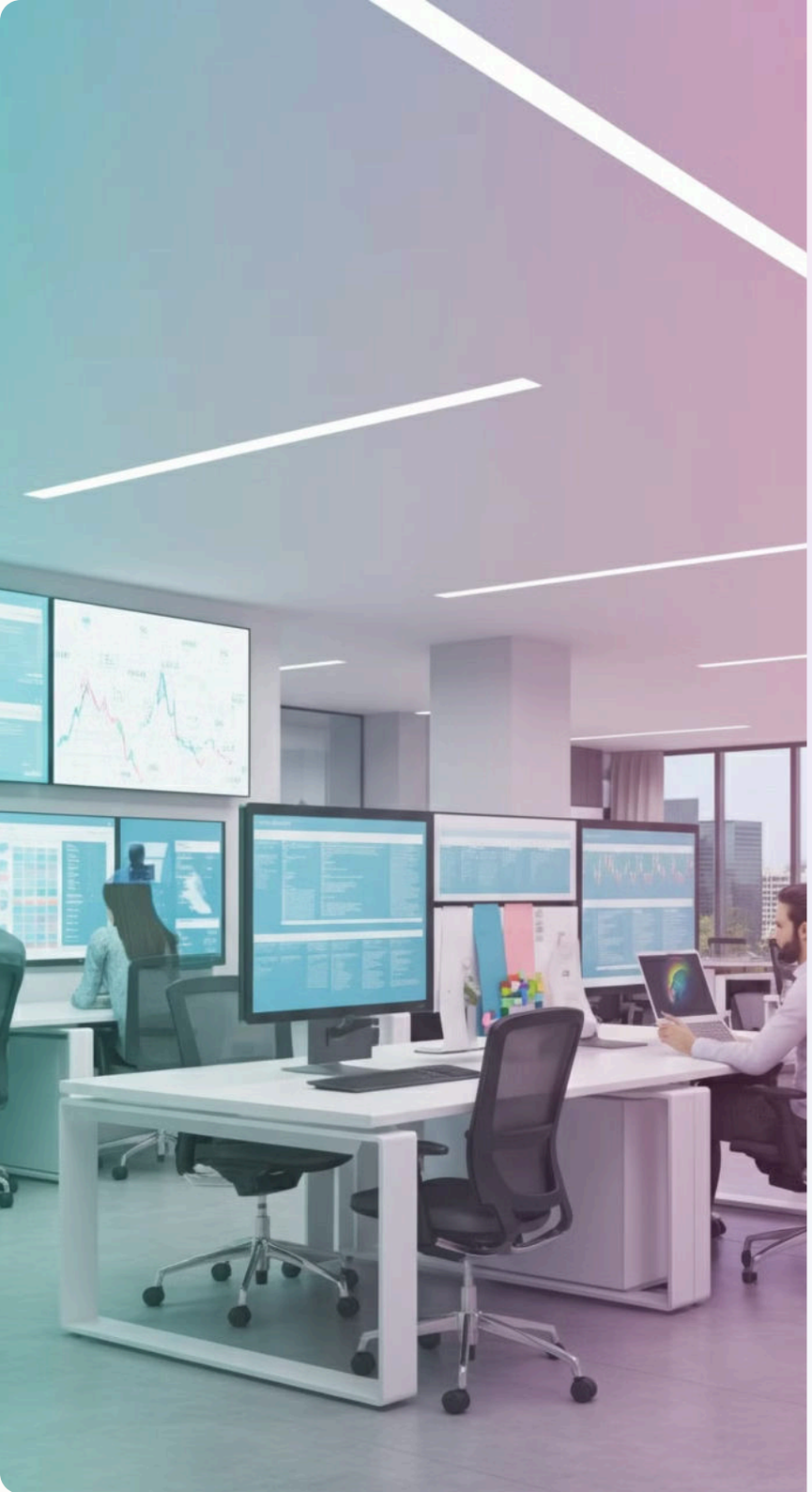


Wilson Financial Database Design: Streamlining Asset and Client Management

Wilson Financial, a prominent financial services company, sought to revolutionize its management of client assets, transactions, billings, and compliance. Our team of experts was tasked with designing a comprehensive database system to address these critical aspects of their operations. This presentation outlines our innovative solution, detailing the intricate business rules, carefully considered assumptions, and the resulting entity-relationship diagram that forms the backbone of this transformative system.

Through our meticulous design and implementation, Wilson Financial is now poised to streamline its operations, enhance client satisfaction, and ensure robust compliance with regulatory requirements. Join us as we explore the intricacies of this game-changing database solution that's set to redefine financial service management.

Project Group 3 Team

Harrison Futch

Shayna Solomon

Understanding Wilson Financial's Needs

1 Comprehensive Client Management

Wilson Financial required a system that could handle both individual and corporate clients, managing their unique identifiers, personal details, and total asset values. The system needed to accommodate multiple assets and transactions per client, ensuring a holistic view of each client's financial portfolio.

2 Robust Asset and Transaction Tracking

The ability to meticulously track and manage assets and transactions was paramount. Each asset and transaction needed a unique identifier, with assets linked to specific clients and transactions including crucial details such as date, amount, and type.

3 Efficient Billing and Employee Management

The system had to support a sophisticated billing structure based on client transactions and assets. Additionally, it needed to manage employee information, roles, and responsibilities effectively to ensure smooth operations and clear accountability.

4 Stringent Compliance Adherence

Given the regulatory environment of financial services, the system had to be designed with compliance at its core. This included features to maintain records, conduct audits, and ensure all activities met SEC requirements.

Crafting the Business Rules

1

Client Management

We established a robust client management system with unique client IDs, comprehensive personal details, and the ability to handle both individual and corporate clients. This system allows for multiple assets and transactions to be associated with each client, providing a complete financial overview.

2

Asset and Transaction Management

Our design incorporates unique identifiers for each asset and transaction, ensuring accurate tracking and management. Assets are linked to specific clients and include important details such as type, value, and acquisition date. Transactions are meticulously recorded with client ID, date, amount, and type.

3

Employee and Billing Management

The system includes a comprehensive employee management module with unique employee IDs, detailed information, and specific role assignments. The billing management system is designed to generate accurate billing records based on client transactions and assets, with each billing record having a unique identifier.

4

Compliance Integration

To ensure adherence to SEC regulations, we integrated robust compliance features. These allow the compliance manager to maintain necessary records, conduct regular audits, and ensure all company activities meet regulatory requirements.

Key Assumptions Guiding Our Design

Client Diversity

Our design assumes a diverse client base, including both individuals and corporate entities. This flexibility allows Wilson Financial to cater to a wide range of clients without compromising on data integrity or management capabilities. We've ensured that the system can handle the unique requirements of each client type efficiently.

Transaction and Asset Complexity

We've designed the system to handle a wide variety of transaction types and asset classes. This assumption allows for future scalability and ensures that Wilson Financial can manage complex financial portfolios without needing significant system changes. The system is prepared to handle everything from simple cash deposits to complex derivative transactions.

Regulatory Landscape

Our design assumes a dynamic regulatory environment. We've built in flexibility to adapt to changing SEC requirements and other financial regulations. This forward-thinking approach ensures that Wilson Financial can remain compliant without major system overhauls, even as the regulatory landscape evolves.

Generating Actionable Reports

1

Client Transaction Summary

This report provides a comprehensive overview of each client's transactions. It includes transaction dates, amounts, and types, offering a clear picture of the client's financial activity. This summary is crucial for account managers to track client behavior and identify potential investment opportunities or areas of concern.

2

Asset Management Report

Detailing the assets managed for each client, this report includes asset types, values, and acquisition dates. It's an essential tool for financial advisors to assess portfolio diversity and performance, enabling them to make informed recommendations for asset allocation and rebalancing strategies.

3

Employee Role Summary

This report outlines the roles and responsibilities of each employee, including their contact information. It's invaluable for management to ensure proper task allocation, identify skill gaps, and maintain clear communication channels within the organization.

4

Billing Summary Report

Summarizing billing records for each client, this report includes billing dates, amounts, and types. It's crucial for finance teams to track revenue, ensure accurate invoicing, and identify any discrepancies or patterns in billing that may require attention or optimization.

Future-Proofing Wilson Financial

Scalability

Our database design is built to scale, accommodating Wilson Financial's growth. As the client base expands and transaction volumes increase, the system will maintain its performance and efficiency, ensuring seamless operations well into the future.

Adaptability

The modular nature of our design allows for easy integration of new financial products or services. Whether it's incorporating cryptocurrency assets or adapting to new regulatory requirements, the system is flexible enough to evolve with the changing financial landscape.

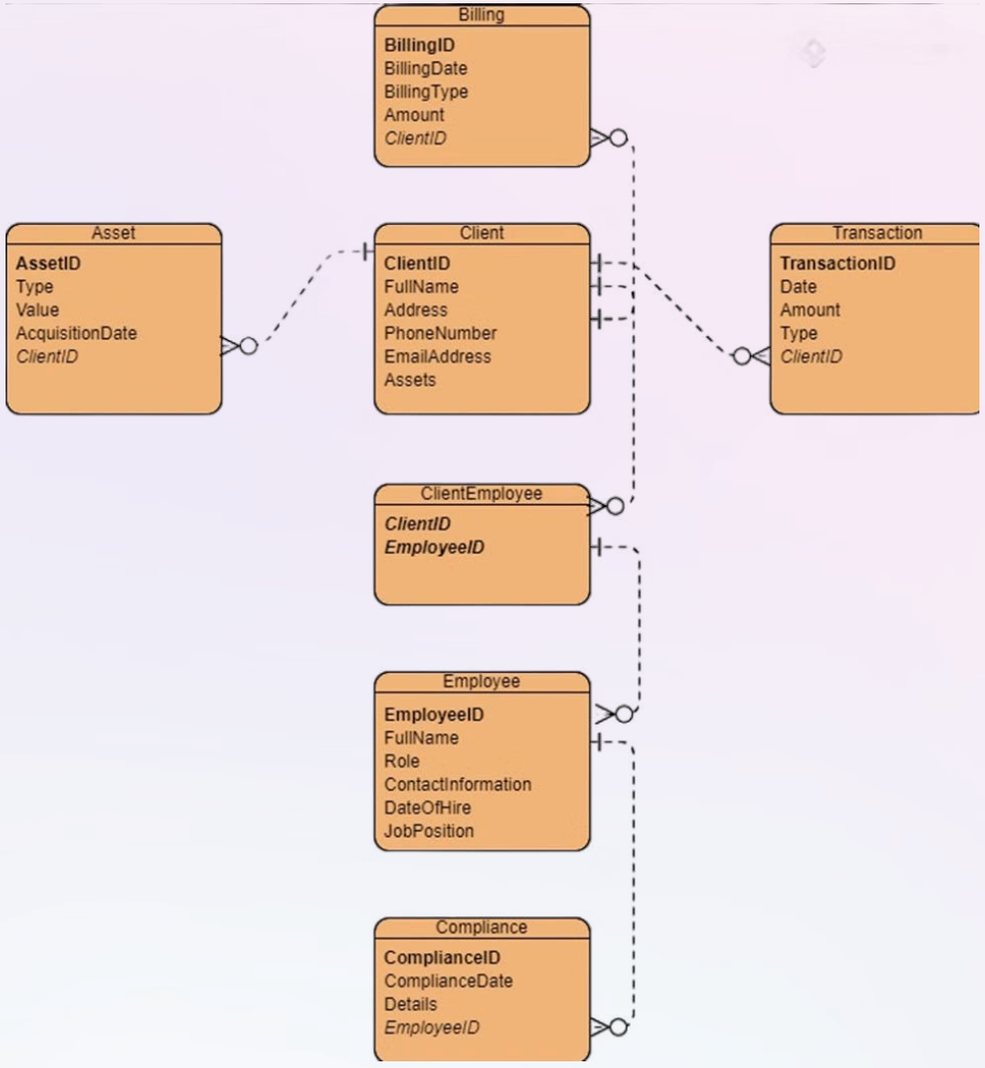
Data Security

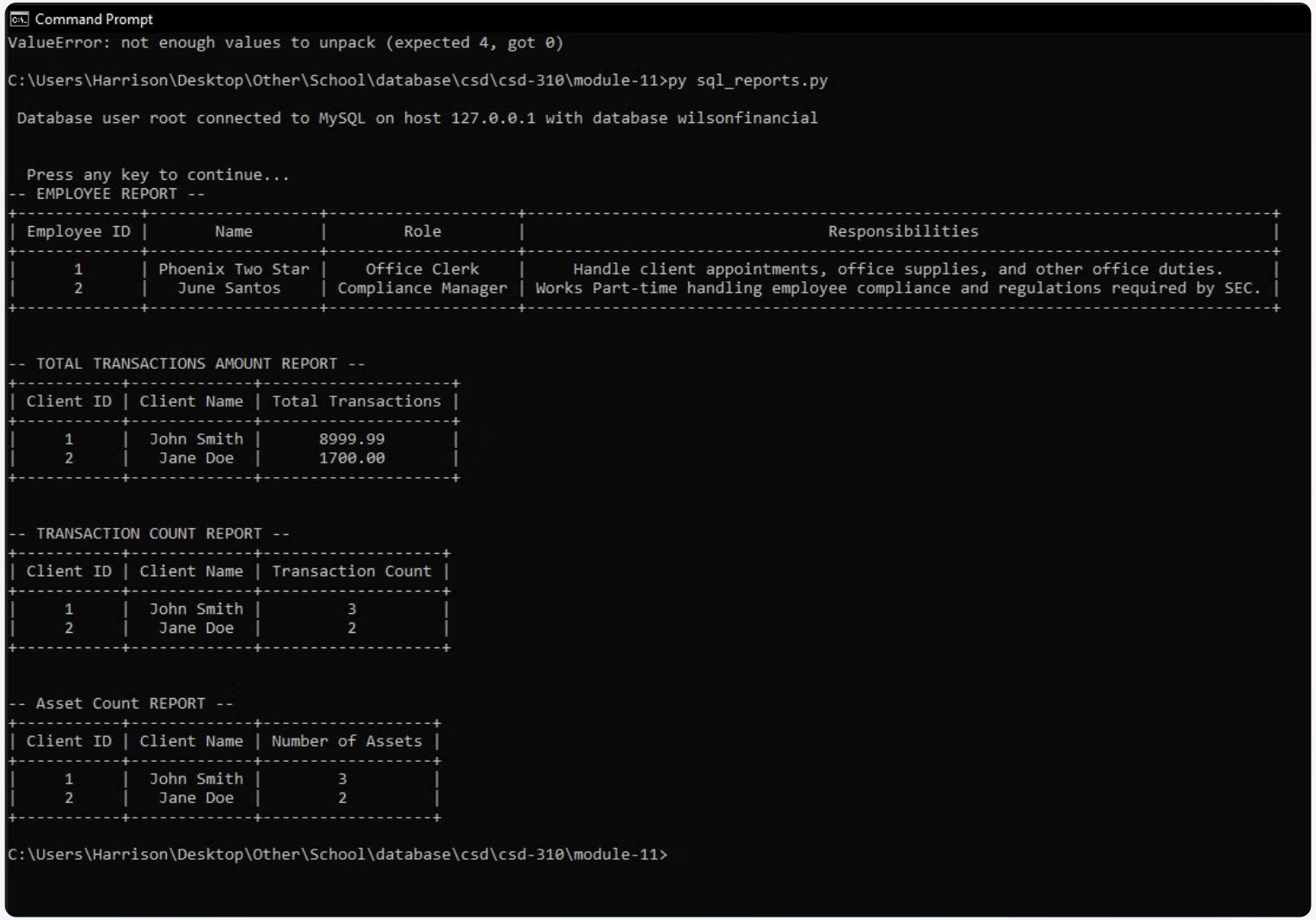
With increasing cyber threats, our system incorporates state-of-the-art security measures. Regular security updates and audits are built into the design, ensuring that Wilson Financial's sensitive financial data remains protected against emerging threats.

AI Integration

The database is designed with future AI integration in mind. As AI technology advances, Wilson Financial can easily incorporate predictive analytics, automated compliance checks, and personalized client recommendations, staying at the forefront of financial technology.

Entity Relationship Diagram




Command Prompt
ValueError: not enough values to unpack (expected 4, got 0)
C:\Users\Harrison\Desktop\Other\School\database\csd\csd-310\module-11>py sql_reports.py
Database user root connected to MySQL on host 127.0.0.1 with database wilsonfinancial
Press any key to continue...
-- EMPLOYEE REPORT --
Employee ID	Name	Role	Responsibilities
1	Phoenix Two Star	Office Clerk	Handle client appointments, office supplies, and other office duties.
2	June Santos	Compliance Manager	Works Part-time handling employee compliance and regulations required by SEC.
-- TOTAL TRANSACTIONS AMOUNT REPORT --
Client ID	Client Name	Total Transactions
1	John Smith	8999.99
2	Jane Doe	1700.00
-- TRANSACTION COUNT REPORT --
Client ID	Client Name	Transaction Count
1	John Smith	3
2	Jane Doe	2
-- Asset Count REPORT --
Client ID	Client Name	Number of Assets
1	John Smith	3
2	Jane Doe	2
C:\Users\Harrison\Desktop\Other\School\database\csd\csd-310\module-11>

db_info_insert.sql

-- if database exists, drop it DROP USER IF EXISTS 'wilson_user'@'localhost';

-- create wilson_user and grant all privileges to the new database -- grant all privileges to the wilson financial database to user wilson_user on localhost CREATE USER 'wilson_user'@'localhost' IDENTIFIED BY 'secure_password'; GRANT ALL PRIVILEGES ON wilsonfinancial.* TO 'wilson_user'@'localhost'; FLUSH PRIVILEGES;

USE WilsonFinancial;

/* ADD RECORDS */

/* Clients */ INSERT INTO Clients (ClientID, Name, ContactInfo, EmployeeID) VALUES(1, 'John Smith', "**johnsmith123@gmail.com**", NULL);

INSERT INTO Clients (ClientID, Name, ContactInfo, EmployeeID) VALUES(2, 'Jane Doe', "**janedoe456@gmail.com**", NULL);

/* Assets */

INSERT INTO Assets (AssetID, AssetType, AssetValue, AcquisitionDate, ClientID) VALUES(1, 'Painting', 5000.0, '2023-05-25', 1);

INSERT INTO Assets (AssetID, AssetType, AssetValue, AcquisitionDate, ClientID) VALUES (2, 'Computer', 1999.99, '2024-01-12', 1);

INSERT INTO Assets (AssetID, AssetType, AssetValue, AcquisitionDate, ClientID) VALUES (3, 'Desk', 500.0, '2024-04-08', 2);

INSERT INTO Assets (AssetID, AssetType, AssetValue, AcquisitionDate, ClientID) VALUES (4, 'Painting', 1200.0, '2023-12-31', 2);

INSERT INTO Assets (AssetID, AssetType, AssetValue, AcquisitionDate, ClientID) VALUES (5, '4k-TV', 2000.0, '2024-06-12', 1);

/* Transactions */ INSERT INTO Transactions (TransactionID, TransactionAmount, TransactionDate, ClientID) VALUES (1, 500.0, '2024-07-10', 1);

INSERT INTO Transactions (TransactionID, TransactionAmount, TransactionDate, ClientID) VALUES (2, 1200.0, '2024-07-15', 2);

INSERT INTO Transactions (TransactionID, TransactionAmount, TransactionDate, ClientID) VALUES (3, 700.0, '2024-07-20', 1);

INSERT INTO Transactions (TransactionID, TransactionAmount, TransactionDate, ClientID) VALUES (4, 200.0, '2024-07-25', 2);

/* Employees */ INSERT INTO Employees (EmployeeID, Name, Role, Responsibilities) VALUES (1, 'Michael Scott', 'Manager', 'Manage operations');

INSERT INTO Employees (EmployeeID, Name, Role, Responsibilities) VALUES (2, 'Dwight Schrute', 'Sales', 'Handle client sales');

INSERT INTO Employees (EmployeeID, Name, Role, Responsibilities) VALUES (3, 'Pam Beesly', 'Receptionist', 'Manage front desk');

INSERT INTO Employees (EmployeeID, Name, Role, Responsibilities) VALUES (4, 'Jim Halpert', 'Sales', 'Handle client sales');

/* Billing */ INSERT INTO Billing (BillingID, ClientID, BillingDate, BillingAmount, BillingType) VALUES (1, 1, '2024-08-01', 150.0, 'Monthly Fee');

INSERT INTO Billing (BillingID, ClientID, BillingDate, BillingAmount, BillingType) VALUES (2, 2, '2024-08-01', 300.0, 'Monthly Fee');

Sql_reports.py

```
import mysql.connector
from mysql.connector
import errorcode
import prettytable
from prettytable import PrettyTable:
# Reference(s) for PrettyTable:
#   https://tech.joellemena.com/python/python-print-sql-query-results-as-table/
#   https://www.geeksforgeeks.org/creating-tables-with-prettytable-library-python/

config = {
    "user": "root",
    "password": "7#x2X_p?q35x6!X",
    "host": "127.0.0.1",
    "database": "wilsonfinancial",
    "raise_on_warnings": True
}

try:
    db = mysql.connector.connect(**config)

    print("\n Database user {} connected to MySQL on host {} with database {}".format(config["user"],
config["host"], config["database"]))
    input("\n\n  Press any key to continue...")

    cursor = db.cursor()

    ## Report 1: List of Employees and their Roles and Responsibilities
##
    query1 = "SELECT EmployeeID, Name, Role, Responsibilities FROM Employees;"


    ## Report 2: Client total Transaction Amount ##
    query2 = "SELECT c.ClientID, c.Name, SUM(t.TransactionAmount) AS TotalTransactionAmount FROM Clients c
JOIN Transactions t ON c.ClientID = t.ClientID GROUP BY c.ClientID, c.Name ORDER BY TotalTransactionAmount
DESC;"

    # ## Report 3: Clients with High Transaction Volume ##
    query3= "SELECT c.ClientID, c.Name, COUNT(t.TransactionID) AS TransactionCount FROM Clients c JOIN
Transactions t on c.ClientID = t.ClientID GROUP BY c.ClientID, c.Name HAVING TransactionCount < 10 ORDER
BY TransactionCount DESC;"

    ## Report 4: Number of Assets each client has Aquired ##
    query4 = "SELECT a.ClientID, c.Name, COUNT(a.AssetID) AS NumberOfAssets FROM Assets a JOIN Clients c
ON a.ClientID = c.ClientID GROUP BY a.ClientID, c.Name ORDER BY NumberOfAssets DESC;"

    employeeTable = PrettyTable()
    transactionsTable = PrettyTable()
    countTable = PrettyTable()
    assetTable = PrettyTable()

    cursor.execute(query1)
    employeeReport = cursor.fetchall()

    ## EMPLOYEE REPORT ##
    print("-- EMPLOYEE REPORT --")
    employeeTable.field_names = ["Employee ID", "Name", "Role", "Responsibilities"]
    for employee in employeeReport:
    employeeTable.add_row(employee)

    print(employeeTable)
    print("\n")

    ## TOTAL TRANSACTION AMOUNT REPORT ##
    cursor.execute(query2)
    totalReport = cursor.fetchall()

    print("-- TOTAL TRANSACTIONS AMOUNT REPORT --")
    transactionsTable.field_names = ["Client ID", "Client Name", "Total Transactions"]
    for total in totalReport:
        transactionsTable.add_row(total)

    print(transactionsTable)
    print("\n")

    ## TOTAL TRANSACTION COUNT REPORT ##
    cursor.execute(query3)
    countReport = cursor.fetchall()

    print("-- TRANSACTION COUNT REPORT --")
    countTable.field_names = ["Client ID", "Client Name", "Transaction Count"]
    for count in countReport:
        countTable.add_row(count)

    print(countTable)
    print("\n")

    ## Total numnber of assets per client ##
    cursor.execute(query4)
    assetReport = cursor.fetchall()

    print("-- Asset Count REPORT --")
    assetTable.field_names = ["Client ID", "Client Name", "Number of Assets"]
    for asset in assetReport:
        assetTable.add_row(asset)

    print(assetTable)
except mysql.connector.Error as err:
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
        print(" The supplied username or password are invalid")
    elif err.errno == errorcode.ER_BAD_DB_ERROR:
        print(" The specified database does not exist") else:
            print(err)

finally:
    cursor.close()
    db.close()
```