# Real Estate Property Management Database

# ****1. Introduction****

## ****1.1 Project Overview****

This report aims to build a Real Estate Property Management Database for a hypothetical company known as Berealty, situated in Berlin. Its main purpose is to propose and develop a database system that will significantly improve Berealty’s organizational structure, targeting increased effectiveness in real estate management and client communications and improving transaction flows. The database system is designed to address the needs of an expanding real estate company, allowing the centralization of multiple aspects of the business, such as property listing, client details, agents, and transactions. It makes sure that organizations place all pertinent knowledge within one coherent great database that one may use when making decisions and business intelligence.

## ****1.2 Learning Objectives****

The purpose of this project is to demonstrate the learner's ability to:

* Knowledge of what DBMS has to do with it and the principles of database design, administration, and management.
* The ideas used in querying and retrieving data from a MySQL database based on business orientation must be implemented.
* Create complex SQL statements to manage and analyze large amounts of data, which may help to understand the functioning of a real estate agency. This includes writing optimized queries for the firm's dealings with property management, client handling, transactions, and business intelligence reports.

# ****2. Phase 1: Database Design for a Real Estate Agency****

## ****2.1 Scope and Objectives of the Database****

This database is intended to be utilized by Berealty, an agency dealing in the real estate business in Berlin, to manage several properties, clients, and sales. The main subjects covered in the database constitution consist of properties, clients, agents, and real estate transactions. The idea is to have a central point whereby all organizational activities, including property listing, client sign-up, agent allocation, and transaction processing, can be easily accomplished. It also must allow business analytics so the database can be queried for transactions, property statuses, and agent performance reports. Several factors that define a system include scalability and data integrity to determine expansion to add access to new clients while eliminating problems connected with data inconsistency.

## ****2.2 Entity-Relationship (ER) Diagram****

An Entity-Relationship (ER) Diagram is a technique that presents the structure of the database used. It assists in showing how the aspects of a system—entities (which can be objects or concepts)—are associated. In database design, entities are real-life objects, while relationships portray the links between these entities. ER diagrams are used to plan the structure of a database so all necessary aspects are covered and data can be easily stored and searched.

Considering the Real Estate Property Management Database of Berealty, the ER diagram consists of entities such as Property, Client, Agent, and Transaction and their attributes or/and relations. These relationships' articulation ensures that data interactions—for instance, how the agents relate to properties and how the clients relate to transactions—are well-defined and well-structured in the database, as seen from the ER diagram.

### Entries, Slot, and Relations

* Property symbolizes the property and real estate under Berealty's management. It includes PropertyID, Address, type, residential or commercial, Price, and status.
* Client: To these stakeholders, Berealty is a company that works to help them achieve their purposes. ClientID, Name, ContactInfo, and ClientType (buyer or tenant).
* Agent: This stands for agents of Berealty Company. It includes AgentID, Name, Contact information, and Commission.
* Transaction: This stands for the account information of transactions between clients and Berealty. Variables such as Transaction ID, Date, Property ID, Client ID, Selling-Agent ID, Selling Price, and Status can be ascribed to attributes.

The ER diagram helps define the nature of these entities and their interactions, ensuring that, to meet Berealty's requirements, the nature of relationships between them, such as one-to-many, many-to-many, etc., is properly defined.

### Participation and Cardinalities

* Client and Transaction: A client can transact with the system more than once, but a transaction possesses information from one client at a time (One-to-Many).
* Agent and Transaction: In one transaction, an agent can deal with many other transactions, although one transaction is dealt only with one agent (One to Many).
* Property and Transaction: A property can be bought and sold in different transactions at different times, and each transaction is unique and relates to a particular property (one to many).

## ****2.3 Normalization of the Database****

Database normalization is a procedure in which data are placed in a database to conform to the least redundancy and composed of distinct groups for each data element. This entails subdividing the database into tables and developing standard forms to minimize the difficulties and replication when dealing with an involved data system for servicing, querying, and updating an extensive database.

Normalization is usually performed based on standard forms (NF) levels. The most common stages are:

* First Normal Form (1NF): The standard checks that every attribute is atomic and every table possesses an attribute that acts as a primary key.
* Second Normal Form (2NF): This form closely monitors to ensure that all other attributes entirely depend upon the key attribute.
* Third Normal Form (3NF): This form enforces the concept that the key determines all the attributes, not any other icon attribute.

In Berealty, the Real Estate Property Management Database, it is normalized up to the Third Normal Form (3NF) to minimize data redundancy. This ensures that:

* Client, Agent, Property, and Transaction details have tables to avoid replicating the data. For instance, the client's information is stored in another table while the Transaction table is linked using a ClientID. This way, there is no need to enter client details in each record of specific transactions.
* The Transaction table consists only of extra data related directly to the particular transaction, including TransactionID, PropertyID, ClientID, AgentID, and the amount of the transaction; other fields, such as information on the property or the client, are located in their tables as foreign keys.

## ****2.4 Design Rationale and Explanation****

In this design, the Real Estate Property Management Database was developed as a scalable, pliable, and accurate database. It takes the properties, clients, agents, and transactions as different tables to minimize the issue of redundant data. The concept of modules enables information to be accessed and changed independently of the other information without negatively affecting the quality of information. They describe real-world business interactions, including agents' roles in effecting transactions, the clients in the contracts, and properties being sold or let out. This normalization of the database to 3NF optimizes the database's query and performs well in business analysis.

# ****3. Phase 2: Database Implementation****

## ****3.1 MySQL Database Setup****

### ****Table Creation****

To implement the database design in MySQL, tables were created for each entity in the ER diagram: Property, Client, Agent, and Transaction. Below are the SQL codes that create the above tables, defining all the necessary data types and constraints.

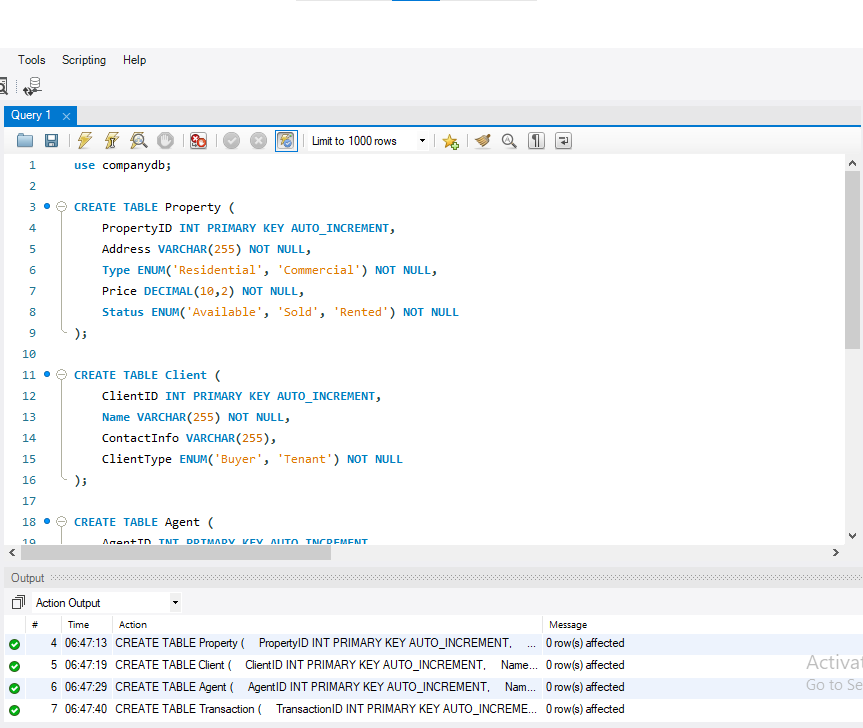


Figure : QL code for table creation in MySQL, including primary keys, data types, and constraints.

### ****Defining Relationships and Constraints****

Foreign vital constraints were used to establish the relationship between the Transaction table and the Property, Client, and Agents table to maintain the integrity of the data. This setup ensures that every transaction is bound to a valid property and the client is a valid agent.

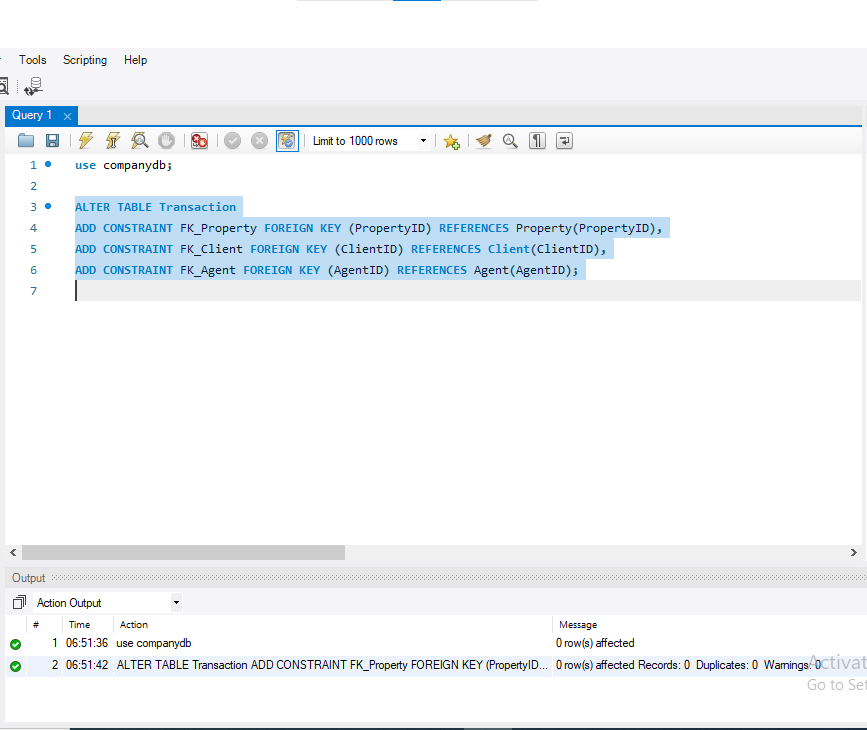


Figure : SQL code defining relationships between the tables using foreign key constraints.

## ****3.2 Sample Data Population****

To explain how the actual database works, sample data representing real estate situations were inputted. These include property data for clients, agents, and transactions, among others.

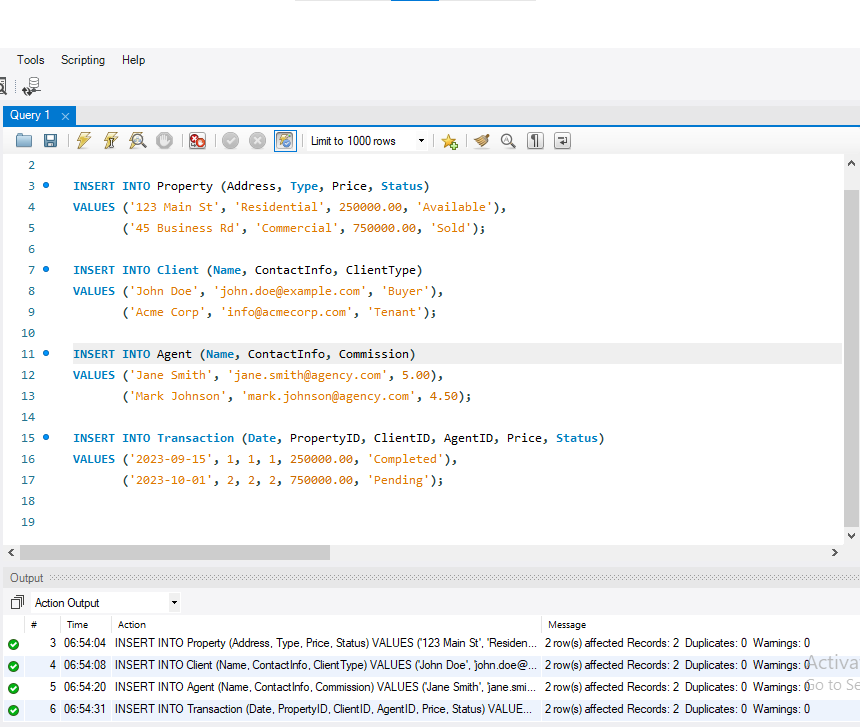


Figure : SQL code for populating the database with sample data for properties, clients, agents, and transactions

## ****3.3 SQL Query Development****

### ****Basic Queries for Property and Client Information****

A basic SQL search can be made to gather some information for basic information properties and clients.

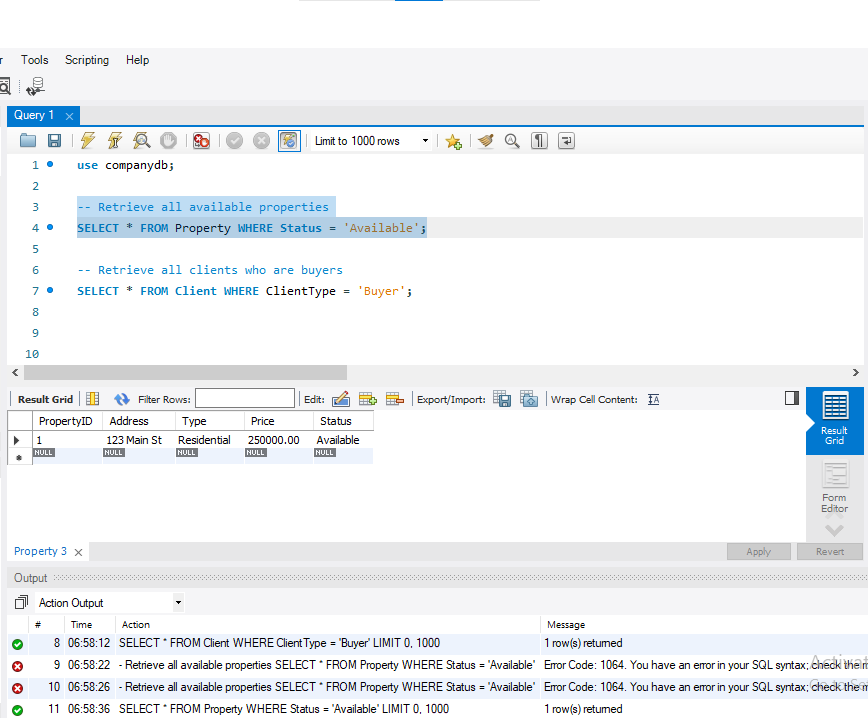


Figure : SQL queries to retrieve property information based on basic criteria.

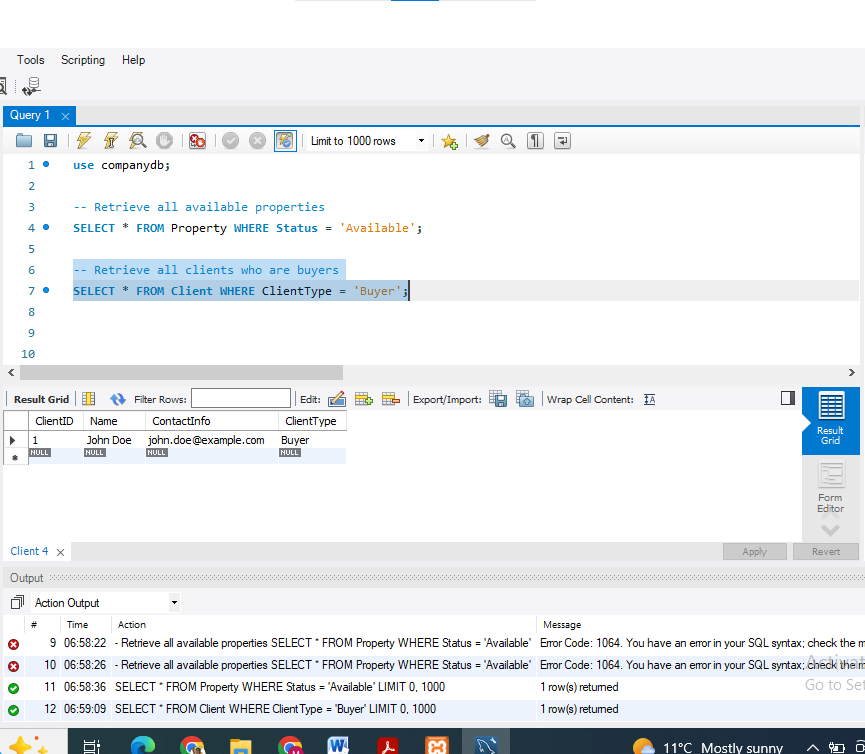


Figure : SQL queries to retrieve client information based on basic criteria.

### ****Advanced Queries with JOIN Operations****

JOIN operations are more robust, allowing for more comprehensive requests for data in more than one table. For example, a query to show transactions managed by each agent:

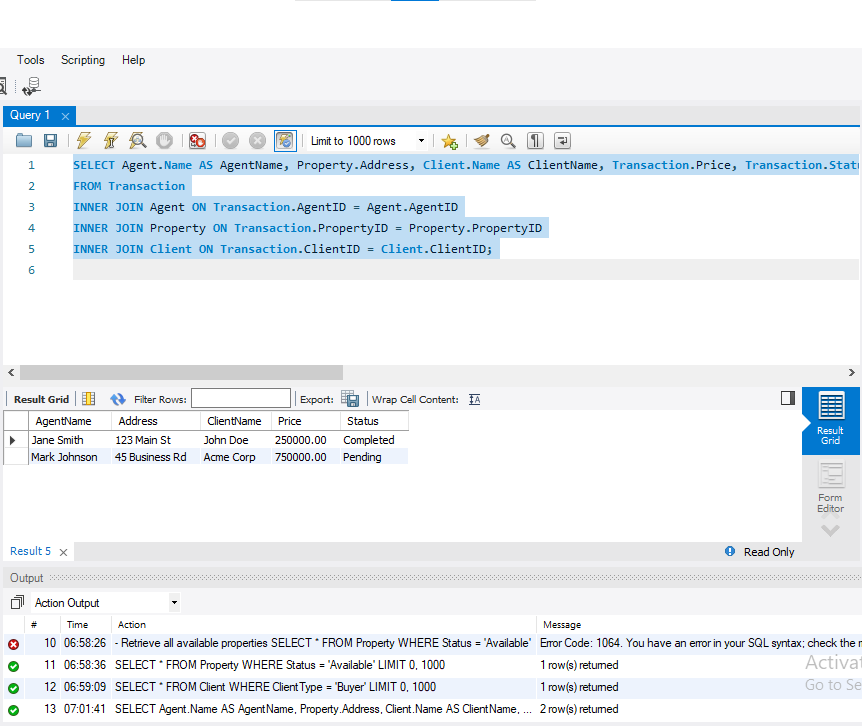


Figure : SQL query using INNER JOIN to retrieve transaction details involving agents, properties, and clients.

### ****Reports and Analytics Queries (Monthly, Quarterly, Yearly)****

The following query produces monthly report of transaction that has been made.

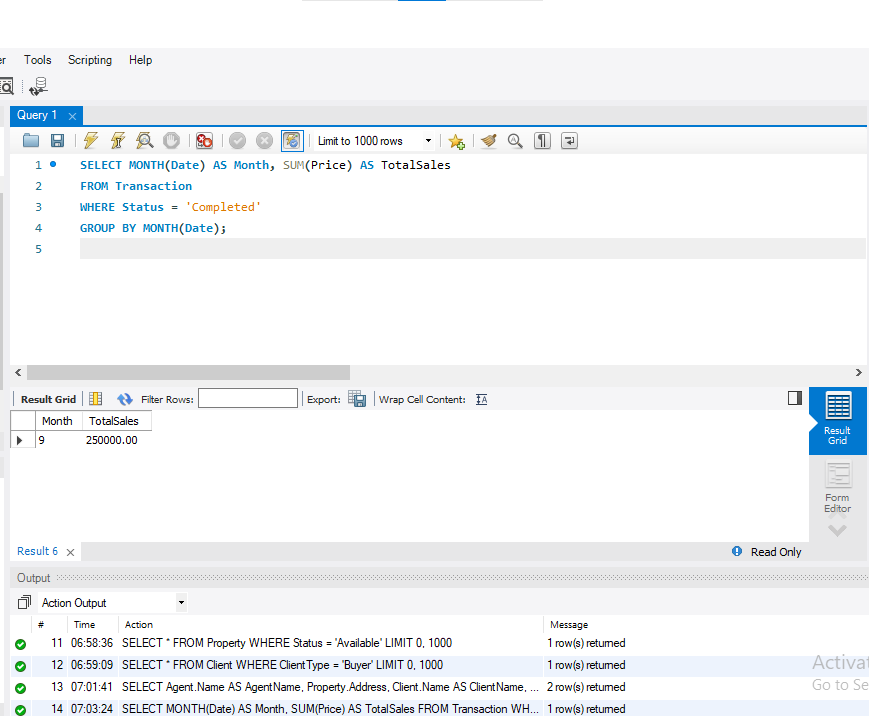


Figure : SQL query to generate a monthly report of completed transactions.

### ****Complex Queries (Sub queries, Aggregates)****

The more complex queries allow for using sub-queries and aggregate functions to give additional information. For example, finding the highest property sale price handled by each agent:

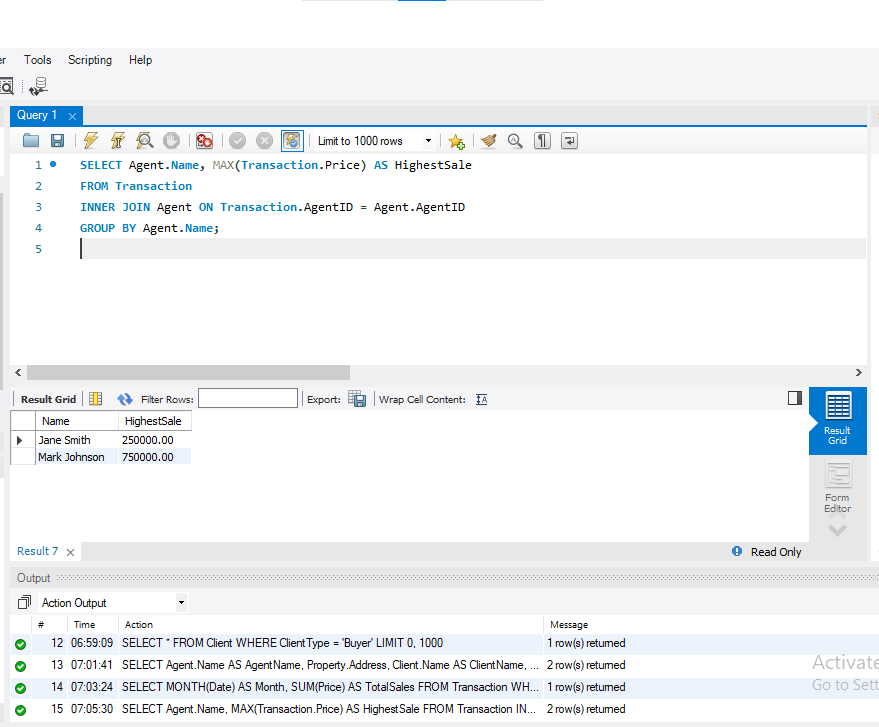


Figure : SQL query using aggregation to determine the highest sale price handled by each agent.

## ****3.4 SQL Triggers****

SQL triggers are procedures that are automatically executed or happily activated when certain events occur in the database. They are intended for specific operations on a table, such as update, insert, or delete operations. Events can also be used for compliance, ensuring data consistency, and executing ad hoc activities and procedures.

Triggers may also be used to prepare automatic actions relative to the Real Estate Property Management Database. For instance, when a transaction concerning a given property is accomplished, the status of the property is modified. For example, if a transaction has an attribute named “Completed,” then it will change the property of another attribute that belongs to the same record from “Available” to “Sold.”

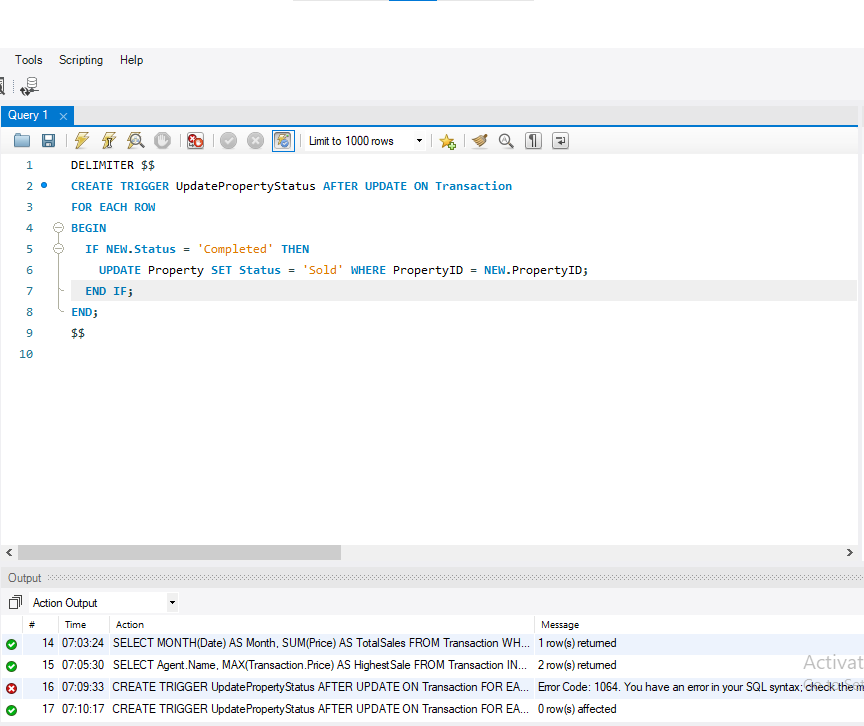


Figure : SQL trigger that automatically updates the property status to 'Sold' when a transaction is completed.

This trigger works as follows:

* It is defined to run following an update on the Transaction table in Table of Contents 15.
* The trigger verifies whether the modified Transaction Status value indicates that the given transaction has been completed.
* If so, then the status of the property with the given PropertyID is changed to “Sold.”

This automation enables statuses for the properties to be kept at a consistent level within the system and will not require any inputs to make them more efficient, not to mention the increased accuracy of data within the system.

# ****4. Phase 3: CAP Theorem Discussion****

## ****4.1 Explanation of CAP Theorem Components****

The CAP theorem, also called Brewer's theorem, is a concept characteristic of distributed computer systems. It states that a database system can only provide two out of the following three guarantees: However, there are three Consistency, Availability, and Partition Tolerance, which have to be achieved in any system. All have different meanings of receiving and dealing with information in the distributed contexts in a system.

### Consistency

Sustainability allows all nodes comprising a distributed database to view the same data simultaneously. Write operations must be made immediately on all nodes—in other words; any read operation should give the most recent value written. While consistency necessarily improves the data's accuracy, it may prove detrimental to its availability under specific conditions.

### Availability

These are more data availability so users can retrieve data even if a node is down, ensured through mirroring. Each request gets a response, either successful or erroneous, but this does not mean the data consistencies are high if consistency replaces temporal updates. In high-availability systems, availability takes precedence, and if needed, some inconsistency of the data may be allowed to be transient in nature.

### Partition Tolerance

The third characteristic is partition tolerance, which determines a system's ability to work. In contrast, a partition is present in the system due to a network partition. Much like the previous scenario, when communication between direct or neighboring nodes is lost, a distributed system must accept a certain number of read-and-write operations where the nodes can no longer access each other. For distributed databases based on different geographical areas, partition tolerance is essential in boosting performance levels.

## ****4.2 Application of CAP Theorem to Real Estate Database System****

Regarding the Real Estate Property Management Database for Berealty, the CAP theorem is best understood when scaling the database system is to be addressed. Suppose the agency chooses to implement a distributed database to manage a more significant number of properties and transactions in different branches or places. In that case, it will also face the problem of the CAP theorem, namely choosing between consistency, availability, or partition tolerance.

Suppose the agency has decided to use new technology and solutions or store information in a distributed database to manage increasing properties and transactions. In that case, this database makes trade-offs between consistency, availability, and partition tolerance.

### For example

* Consistency would create an efficient system by maintaining the alignment of the P2P platforms' listed properties and the status of transactions in all regions. However, availability may be low during network failure because the system has to wait for updates to be propagated to all nodes.
* Availability would enable real estate agents to continue finding information about properties and doing business, given probable system malfunctions. However, this leads to problems, such as the property status showing "Available" in one location and "Sold" in another.
* Partition Tolerance would be mandatory for the geographically distributed real estate market, where the database would be required to work even if the network is under stress. This would have reduced the complete system shutdown during network problems, but depending on Berealty’s business goals, it may entail the compromise of either consistency or availability.

## ****4.3 Relevance of CAP Theorem in System Design and Performance****

The CAP theorem is helpful when deciding on the architecture and system of an extensive and distributed real estate database. Based on their business requirements, Berealty must decide which of these three non-singularities to focus on: consistency, availability, and partition tolerance. For instance, reliability can be given the top priority in confidential activities such as trading, as the data must be valid and current.

* Scalability would be necessary to guarantee that agents and clients could always have access to property data, optimum during busy periods or when there is frequent intermittent connectivity.
* Partition Tolerance is essential for addressing the system's priority and sustainability as well as the database system's evolution and geographical distribution.
* Knowledge of such compromises helps to orient the construction of the system to meet Berealty's needs while providing stability and data protection throughout real estate management.

# ****5. Conclusion****

## ****Summary of Database Design and Implementation****

The Real Estate Property Management Database for Berealty was developed to effectively meet properties, clients, agents, and transaction management functions. The database was developed smoothly, starting with constructing an Entity Relationship (ER) diagram that perfectly depicted the real estate agency's operational needs. A significant factor in the design was a keen focus on data integrity and efficacy, including normalization and appropriate entity associations. These database structures were implemented in MySQL, and tables were created. Relationships and constraints were defined while sample data were loaded to be realistic.

This also entailed establishing enhanced SQL skills in querying, data acquisition, and report generation. Moreover, SQL triggers were used to improve tasks such as changing the status of a property after its transaction occurred. When developing and deploying the database, the key considerations were its suitability for expansion as more operations were added to Berealty.

## ****Key**** Insights ****and Final Thoughts****

This project has brought a close realization of how to properly design a database when performing a series of operations in a real estate setting. Key Insights include:

* ER diagrams play a crucial role in mapping business requirements and presenting the best relational database model.
* The benefits of normalization are cutting down on redundancy and keeping the data as accurate as possible.
* SQL is the work where the basic and complex queries are done, as well as various operations like manipulation, reporting, and automation.
* How SQL triggers have been employed to keep the functioning smooth and ensure that processes that would usually need human input are automated.

## ****Implications****

Successfully implementing this database for Berealty's business processes has several critical implications. The system structure often follows a centralized and organized approach, enhancing decision-making and customer solutions. Automated triggers limit human involvement and change property status as soon as an observation is made. Also, the system's scalability allows the database to grow together with Berealty's company portfolio without affecting the work's efficiency.

As reflected in this project, the simple tweak of improving the database system has the potential to revolutionize Berealty's operations and allow it to meet the current requirements of the Berlin market while focusing on long-term efficiency.

# ****Appendix****

## A. ER Diagram Overview

* Important entities such as Properties, Clients, Agents, and Transactions are developed during the creation of an ER diagram.
* Relationships include allocating properties and tasks to agents, deals between Relational-Plus and its clients, and property management.
* The cardinalities and the participation constraints are made to represent fundamental life interactions.

## B. Database Tables

* There are also eight tables: Property, Client, Agent, and Transaction, with relations between them defined by primary and foreign keys.
* Constraints: The main idea of primary keys is that records cannot be the same; foreign keys aim to connect related material.

## C. SQL Queries Overview

* Queries were developed for:
  + Information for properties and clients alike.
  + Transaction history and performance of the agents.
  + Monthly, quarterly, and yearly reports for analytic work.

## D. Triggers

* Ensures that properties undergo status change notification after a transaction, thereby minimizing interferences.

## E. CAP Theorem Application

* The CAP theorem principles (Consistency, Availability, and Partition Tolerance) were considered while designing the systems discussed in this paper to achieve an optimal combination that prioritizes consistency and availability.