

DBD 271 Project

Group project



Group members:

Marcel De Lange-577610

Realene Repinga – 577171

Ashton Damons - 578008

Table of Contents

[Introduction and background: 2](#_Toc131282253)

[ERD Diagram: 2](#_Toc131282254)

[Normalization: 3](#_Toc131282255)

[Entity Identification and explanation: 3](#_Toc131282256)

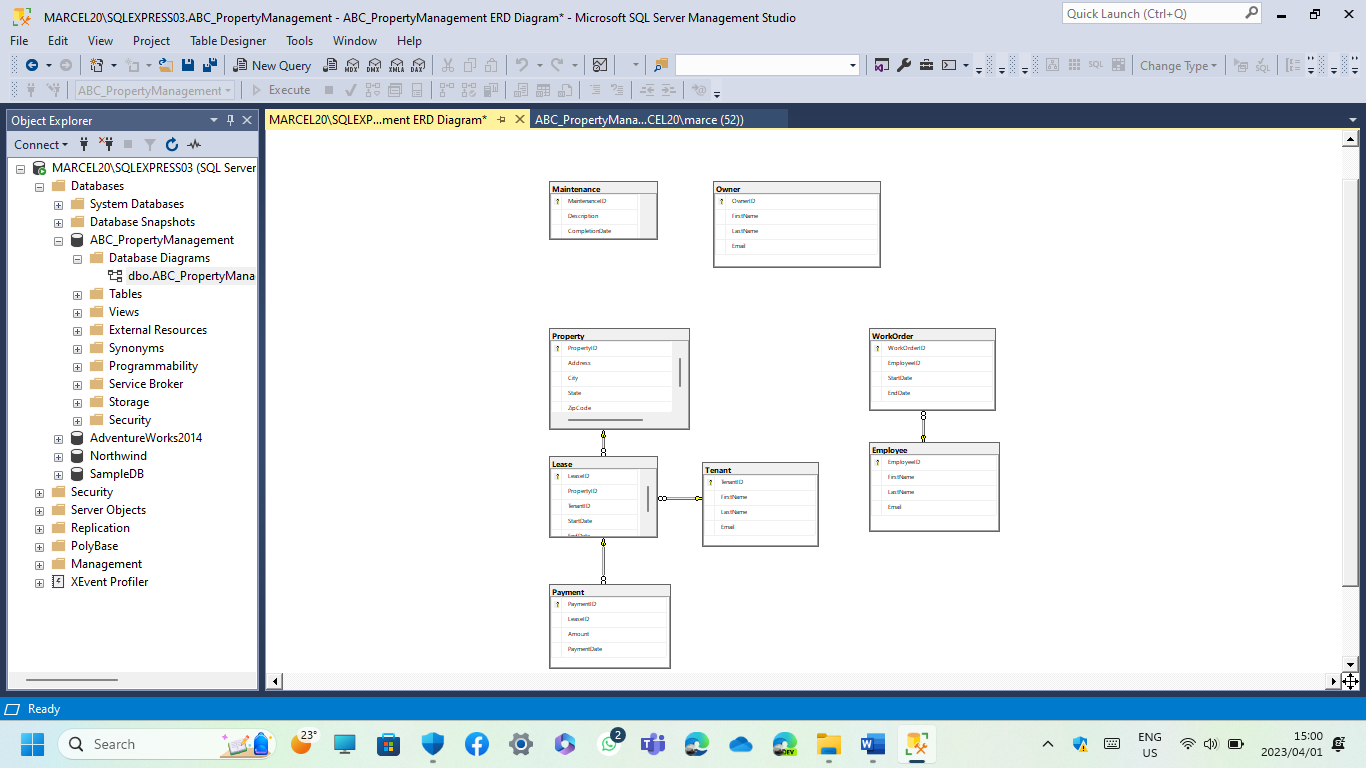
[Queries, Views, and Objects: 4](#_Toc131282257)

# Introduction and background:

ABC Property Management is a professional property management firm dedicated to providing outstanding service to both property owners and tenants. Our experienced team is committed to increasing the value of your property investment while also ensuring that it is well-maintained and profitable.

We understand how difficult and time-consuming it can be to own and manage a property. Property management can be fraught with problems that can quickly become overwhelming, from dealing with difficult tenants to dealing with maintenance issues and collecting rent. Fortunately, our experienced team is here to assist. We specialize in resolving all types of property management issues, from tenant conflicts to property maintenance issues and everything in between. We work tirelessly to ensure that your property is well-maintained, profitable, and trouble-free, so you can focus on what is important to you. With our extensive range of property management services, you can be confident that we have the knowledge and resources to handle any issue that arises.

# ERD Diagram:



# Normalization:

The first step was to make a list of all entities. This database will contain 8 entities. We conducted research on the characteristics that best describe each entity and will aid in the solution of the challenge.

Make a list of all the attributes and double-check that each one has a unique value. Each attribute should only have one value.

• The Primary Keys of each entity have been determined.

• Identified foreign keys and associations that are required by the business.

• Decomposable columns have been eliminated.

• Looked for transitive functional dependencies.

The 3NF is attached on the ERD as it lacks transitive functional dependencies and columns that cannot be decomposed.

# Entity Identification and explanation:

Employees

* PK EmployeeID
* FirstName
* LastName
* Email

Tenant

* PK TenantID
* FirstName
* LastName
* Email

Work Order

* PK WorkOrderID
* FK EmployeeID
* StartDate
* EndDate

Owner

* PK OwnerID
* FirstName
* LastName
* Email

Payment

* PK PaymentID
* FK LeaseID
* Amount
* PaymentDate

Property

* PK PropertyID
* Address
* City
* State
* ZipCode

Maintenance

* PK MaintenanceID
* Description
* CompletionDate

Lease

* PK LeaseID
* FK PropertID
* FK TenantID
* StartDate
* EndDate

# Queries, Views, and Objects:

#### sub query:

* 1. This query returns all properties that are leased by tenants with the first name ‘David’.

#### CTE:

* 1. This query creates a CTE named PropertyCTE that selects all properties in California state.
  2. The second part of the query selects all properties in Anytown from the CTE

#### Join:

* 1. This query will return the PropertyID and Address columns from the Property table and the FirstName and LastName columns from the Tenant table for all rows where there is a match in all three tables.
  2. This can be useful if you want to see which tenants are associated with which properties.

#### View

* 1. This query creates a view called WorkOrderView that contains the columns WorkOrderID, FirstName, LastName, StartDate, and EndDate.
  2. It then selects these columns from the WorkOrder and Employee tables

#### cursor query:

* 1. This example declares a cursor named PropertyCursor that retrieves data from the Property table.
  2. The cursor retrieves all columns from the table and stores them in variables that are declared at the beginning of the script.
  3. The OPEN statement opens the cursor and retrieves the first row of data. The WHILE loop iterates through each row of data returned by the cursor and prints out each column value.
  4. Finally, the CLOSE statement closes the cursor and deallocates it.

#### Case(View):

* 1. This will create a new view called PaymentCaseStatementView that includes the PaymentID, Amount, PaymentDate, and a new column called PaymentAmountCategory that categorizes the payment amount as either ‘High’, ‘Medium’, or ‘Low’ based on the payment amount value.

#### Stored Procedure:

* 1. This stored procedure returns lease details for a specific property ID.
  2. It joins the Lease, Property and Tenant tables to return the lease details along with tenant information.
  3. You can modify this stored procedure to suit your needs by changing the columns returned or adding additional filters.

#### Trigger:

* 1. trigger that fires when you try to insert a new row into the Lease table

#### user defined functions:

* 1. This function returns a single value of the total amount of payments made.

#### LOGIN 1

* 1. For manager, can access all

#### Login 2

* 1. Secretary is only allowed to insert, update, select something from the lease table and by that does not have access to the employee table

#### Backup

* 1. Full back up of database.