Lewis University  
CPSC 50900: Database Systems  
Term Project

Rental Management System

Work products stored in the GitHub repository PUT GITHUB URL HERE

Table of Contents

[Initial Proposal 2](#_Toc85814569)

[Data Sources 2](#_Toc85814570)

[Data Storage Alternatives 2](#_Toc85814571)

[Relational Database Design Process 3](#_Toc85814572)

[Relational Database Design 3](#_Toc85814573)

[Data Definition Language (DDL) Scripts 3](#_Toc85814574)

[Data Manipulation Language Scripts 4](#_Toc85814575)

[Indexes 5](#_Toc85814576)

[Views 5](#_Toc85814577)

[Triggers 5](#_Toc85814578)

[Transactions 5](#_Toc85814579)

[Database Security 6](#_Toc85814580)

[Locking and Concurrent Access 6](#_Toc85814581)

[Backing Up Your Database 6](#_Toc85814582)

[Python Programming 7](#_Toc85814583)

[PHP Programming 7](#_Toc85814584)

[Suggested Future Work 8](#_Toc85814585)

[Activity Log 8](#_Toc85814586)

# Initial Proposal

*Rubric: Your response to each of these six questions will be graded out of 3 points.*

* *3 points: clear, complete descriptions that convey the importance and meaning of your data*
* *2 points: mostly clear descriptions, although some additional data would have helped in some sections*

*1 point: necessary Description: You will describe the data you aim to store. What data will be storing? Why are you interested in this data? Why is it important? Where will the data come from? Who will use this data? What kind of application do you plan to build with it?*

* *details are lacking in many of your responses.*

*You will also earn 2 additional points for coming up with a descriptive title for your project.*

*Total points possible: 20*

**Rental Management System Proposal**

**Rental Management**

As demand for housing increases among tenants, so requires independent house owners to discover solutions that allow tenants to browse their housing plan, choose choices, and pay for their rent online. This new automation will allow housing to increase its reach and push the boundaries of what they're capable of. The rental management system will help find paying guests in facilities availability, workplaces, and homes in major cities. The rental management system revolves around the owners and customers. The apartment, Office, House, and Paying Guest's information is sent to the Owner. The size, rent, and accommodation location are all available to the client. The Rental Management System will be great for rental owners because it saves time and allows them to communicate with the person looking for such facilities, reducing the need to explain room specifics over the phone. The appropriate housing is based on customer requirements and a simple search for a suitable apartment, office, or home. People will save time and money by using the Rental Management System.

**Project Objectives**

The main objective is to create a rental management system that will oversee all rental transactions 24 hours a day, seven days a week. The system will keep track of all transactions and their corresponding attributes. The overarching objective is to provide clients with the greatest digital experience possible; another goal is to save clients time by eliminating the need to visit the office and complete transactions quickly and securely. Customer safety is also at the forefront of the design's goals (Chauhan, 2020). By protecting their accounts, and associated data, the design will maintain the privacy and security of the customer's data. A key objective is to make money transfers, housing bookings, rent payments, and other tasks simple and efficient.

Tenants Files

Landlords and property owners are responsible for maintaining current and past tenants records.  Important tenant files to store are All emails and correspondence with the Tenant, Eviction paperwork and records, correspondence regarding approval, legal notices and corresponding actions, lease violations, and corresponding action records. Deposit records are also useful documents. Other details to store are rent payment records and the signed lease or rental agreement, including any changes or updates added during the tenancy period. Work order or maintenance requests, and details of how and when they were handled

**Written requests for entry**

You should also retain application records, tenant screening, and all correspondence regarding approval or denial of rental applications from all rental applicants even if they never lived at one of your rental properties – retaining rental applicant data will help protect you if an applicant files a discrimination claim.

**The Project design Includes:**  
I will compile a list of requirements, including software programming language such as C# .Net to be used and the database model, i.e., MS SQL Server, and IDE such as Visual Studio and SQL Server Management Studio.

# Data Sources

*Description:* *Gather your data in text files. The text files may be csv, tab-delimited, xml, json, or another custom format. Not all the files need to be of the same type. Identify what each file contains by indicating where it came from, explaining in detail how it structured, and describing how you will reorganize the data into a relational database. Post your data files to your GitHub repository, and provide samples of the data in your Word doc.*

*Rubric: Your work will be graded as follows:*

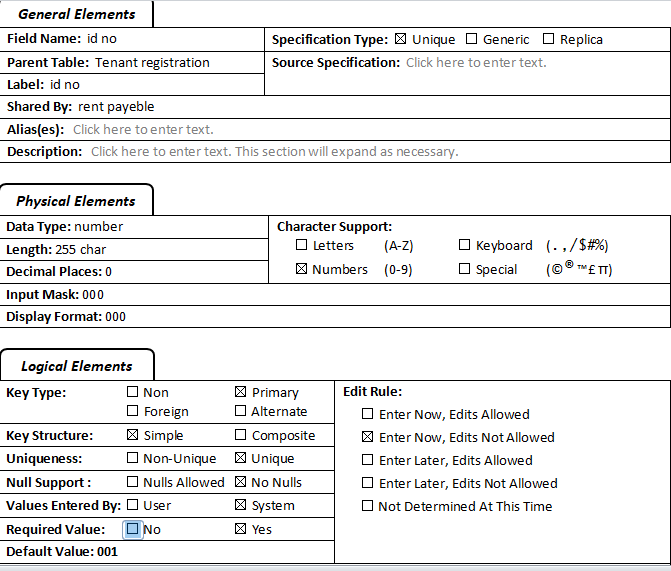
* *5 points: you gathered multiple data files that contain the data that will populate your databases. If you do not use multiple data files, you will not receive credit.*
* *5 points: you described the contents of the data files in detail, including referencing their origin and explaining how they were structured.*
* *3 points: you identify which fields you plan to include in your database, including their data types and any constraints you expect to impose on the data or steps you'll have to take to clean up the data.*
* *2 points: you post the data files to your GitHub account and make it possible for me to see them.*

*Total points possible: 15*

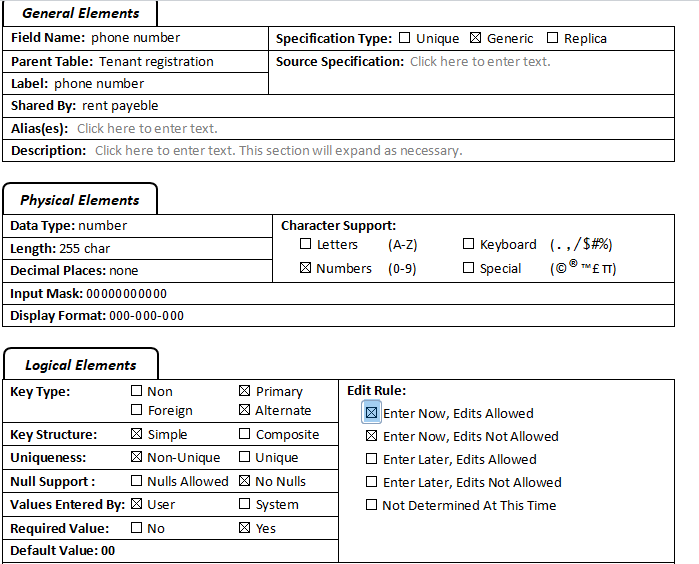
ENTER YOUR DATA SOURCES DESCRIPTION HERE

The data will be collected from the tenants who will have access to a dashboard. They may examine their tenancy data, pay rent and levies online, make maintenance requests, get rental receipts, and connect with landlords—applicants to the login portal. When an applicant wants to rent a property, they may sign a lease or renewal contract—Portal for owners. Access to property information is available online for property owners. Documents such as financial accounts or unit inspection and property reports should also be accessible. Below are the entry fields

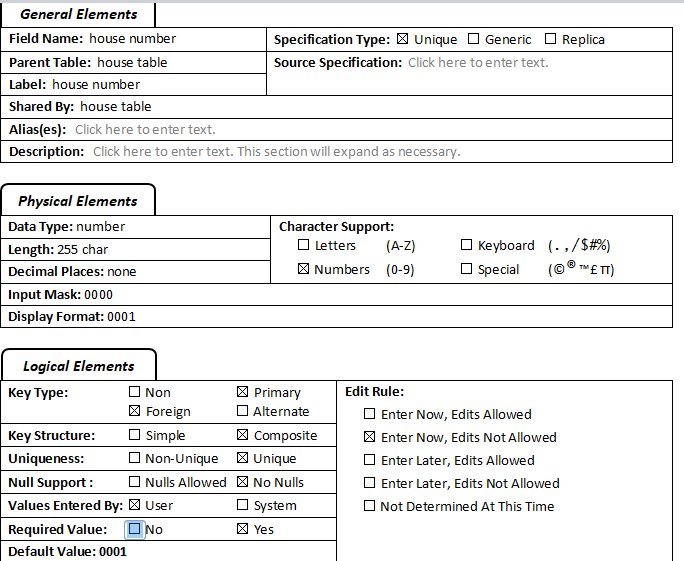
**Tenant registration table**



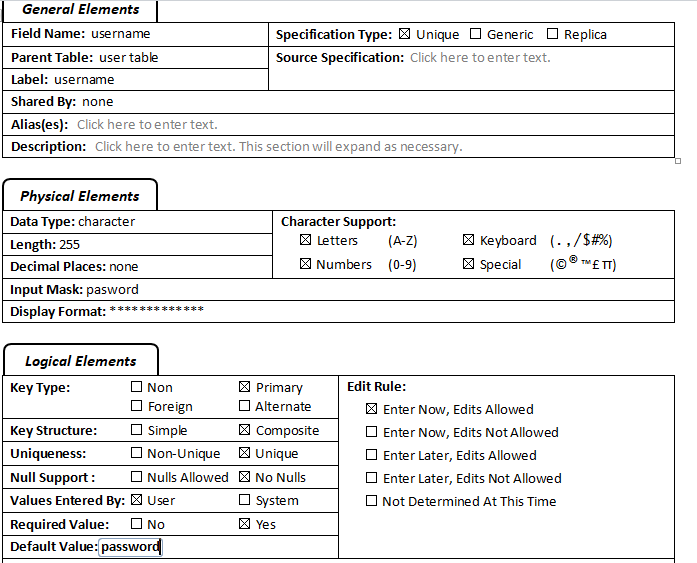
House tables



**Field definition specification for house table**



Tennants table



# Data Storage Alternatives

*Description: We will study alternatives to storing data in a relational database. Some of the alternatives come from several decades ago, including the hierarchical and network models. Some are newer options, such as NoSQL databases that use JSON or some other encoding. Describe in detail how to store the data using two alternatives to relational databases. Be sure to describe how you would implement the alternatives and the advantages and disadvantages of each.*

*Rubric: Your work will be graded as follows*

* *5 points for clearly describing how your data could be stored using one alternative to relational databases and what the advantages and disadvantages of that approach would be.*
* *5 points for clearly describing how your data could be stored using another alternative to relational databases and what the advantages and disadvantages of that approach would be.*

*Total points possible: 10*

ENTER YOUR ALTERNATIVE DATA STORAGE IDEAS HERE

Local Storage

The system will use a cloud-based storage alternative to safely store various files for several years that are conveniently structured, referenced, and accessible. If a calamity occurs, a paper file is likely to be lost or ruined permanently, and it takes up a lot of physical space. It is possible for a computer, server, or storage device to get infected with a virus, be hacked, or otherwise malfunction. There are just too many dangers connected with storing and managing large amounts of data on a hard disk; that is why we will use cloud-based storage.

Online Storage

Github will be widely used in this project to store the codes, and also, we will include a self-hosted site for MySQL database which will be remotemysql.com. Online data storage is widely accepted as the most secure and convenient solution for storing your most essential documents. It is possible to save infinite data safely online with your property management software. Clients will have access to limitless online storage for their key business papers and integrated accounting, property, Tenant, and vendor capabilities that enable clear documentation of vital data. It is possible to upload documents to the application to be saved in the master file library in the Amazon cloud or allocated directly to a tenant or property.

# Relational Database Design Process

*Description: Consider the list of fields you identified in part c. Identify functional dependencies that exist among them. For each functional dependency, identify the determinants and the fields they determine. This becomes the basis for identifying your entity sets, which then become your tables. Give each entity set or table you identify in this way a unique and clear name, making sure that the names you use are singular nouns. Then list the relationships that exist among the various entity sets. For each relationship, identify its connectivity (one-to-one, one-to-many, many-to-many) and participation (optional or mandatory). Finally, make sure that none of the attributes you've assigned to each entity set are multi-valued. If they are, take the steps needed to break them down.*

*Rubric: Your work will be graded as follows:*

* *8 points for identifying all the functional dependencies, including determinants and the columns whose values they determine.*
* *2 points for naming the entity sets that make up your data with clear, easy-to-understand names.*
* *6 points for identify the relationships among the entity sets and identifying connectivity and participation for each.*
* *2 points for breaking down multi-valued attributes.*

*Total points possible: 18*

ENTER YOUR RELATIONAL DATABASE DESIGN DESCRIPTION HERE. INCLUDE SOURCE CODE AND SCREEN SHOTS.

**Relational Database Design**

Database design must be thorough and exacting since it is the heart of the information system. After doing a demand study, the database has to be analyzed and designed. One of the most common database types is the relational database. Data is organized and structured in relational databases throughout the design phase. The main issue is the structure of the relational model. The table structure of the database may be pretty simple to deal with for small databases.

In contrast, as the project's scope has grown, the corresponding database has gotten more sophisticated, and the connection model of the table structure has become more complicated. Incomplete data will cause the table structure to be restructured, resulting in updating the data. Because of this, it is vital to standardize the database architecture, remove duplicate data, increase database storage efficiency, data integrity, and scalability.

The most high-level data model used in database design and the one used in real estate management is based on the concepts of the Entity-Relationship (ER) model. The Entity-Relationship (ER) model. It is a diagrammatic technique that provides a generalized approach to the representation of data, which is particularly suitable and helpful in designing relational database systems. Using the Boyce-Codd Normal Form (BCNF)

When a table has more than one candidate key, anomalies may result even though the relation is 3NF. Boyce-Codd standard form is a particular case of 3NF. A connection is in BCNF if, and only if, every determinant is a candidate key.

The semantic rules (business rules applied to the database) for this table are:

Each estate has many several t in several tenants.

For each Major, a given estate has only one Manager

Each estate has several tenants.

Each Tenant is allocated only one to one estate.

Each Manager collects money from different clients.

The functional dependencies for this table are listed below. The first one is a candidate key; the second is not.

House Id, house location ——>  estate

Estate ——>  Manager

Anomalies for this table include:

Delete – Manager deletes tenants' info

Insert – a new manager needs a tenant

Update – inconsistencies

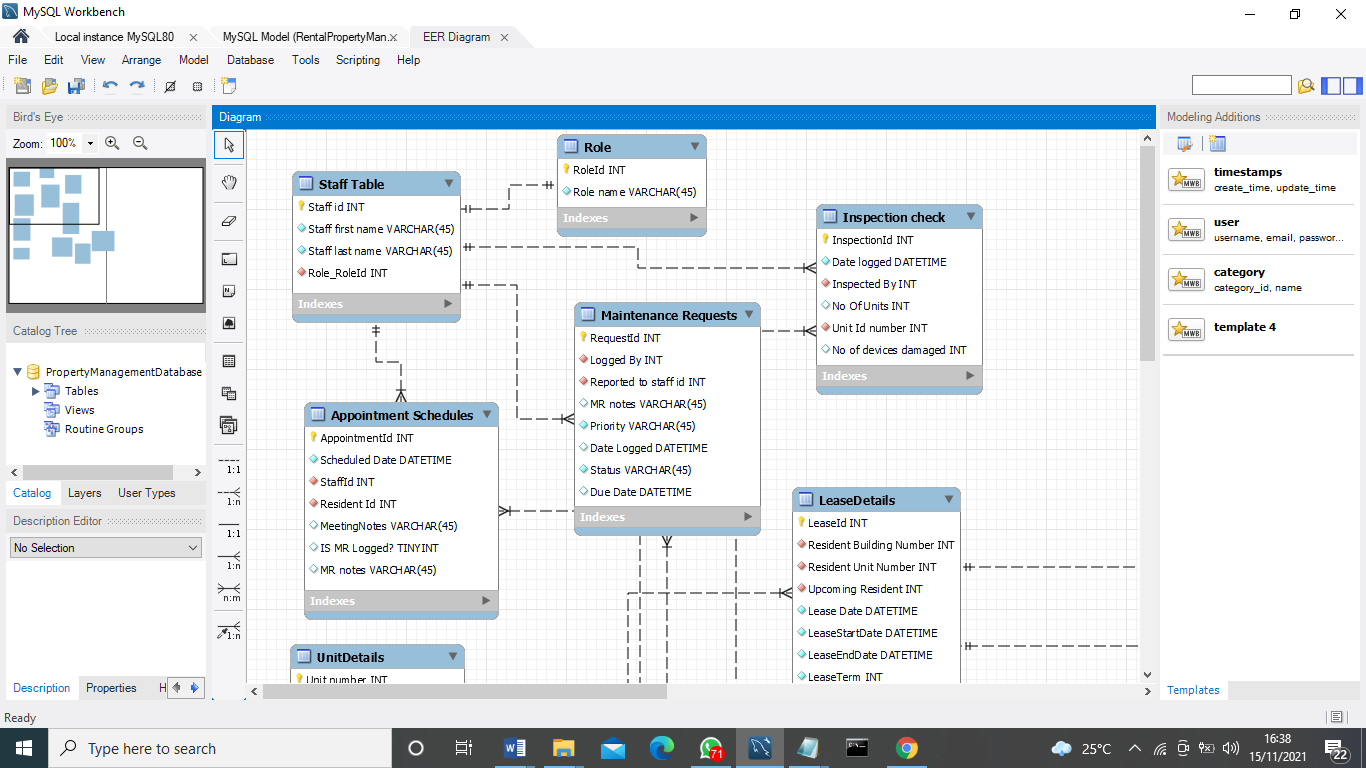
Note: No single attribute is a candidate key.

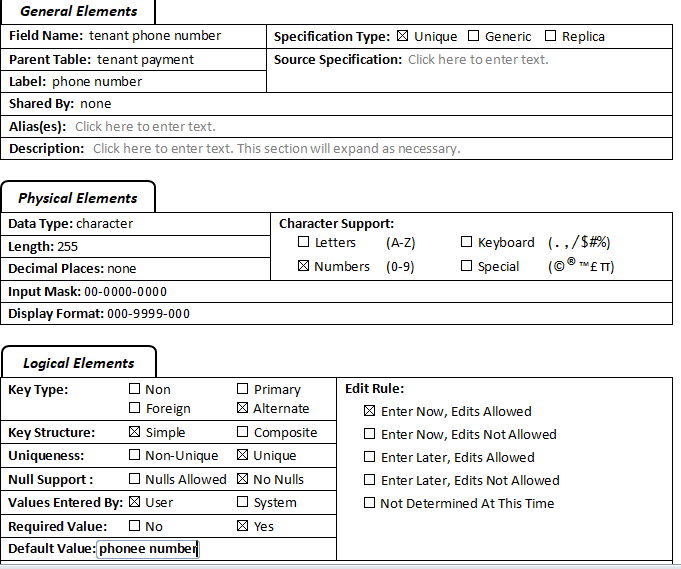
PK can be tenants, Major or housed, Manager.

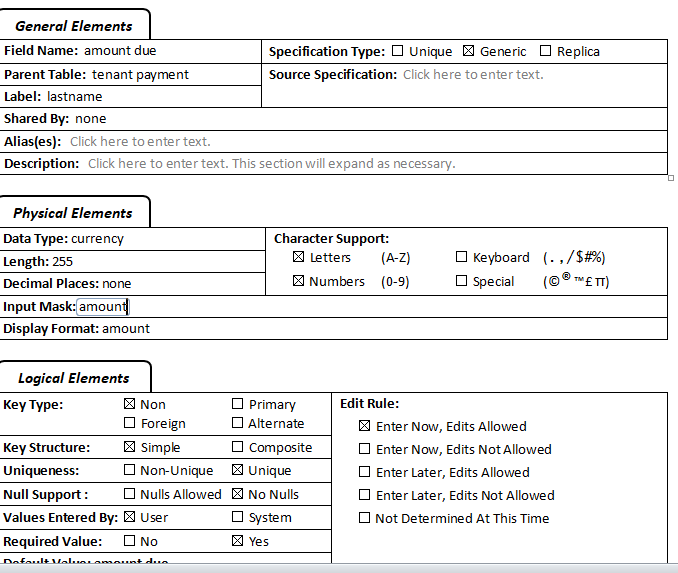
To reduce the relational diagram relation to BCNF, you create two new tables:

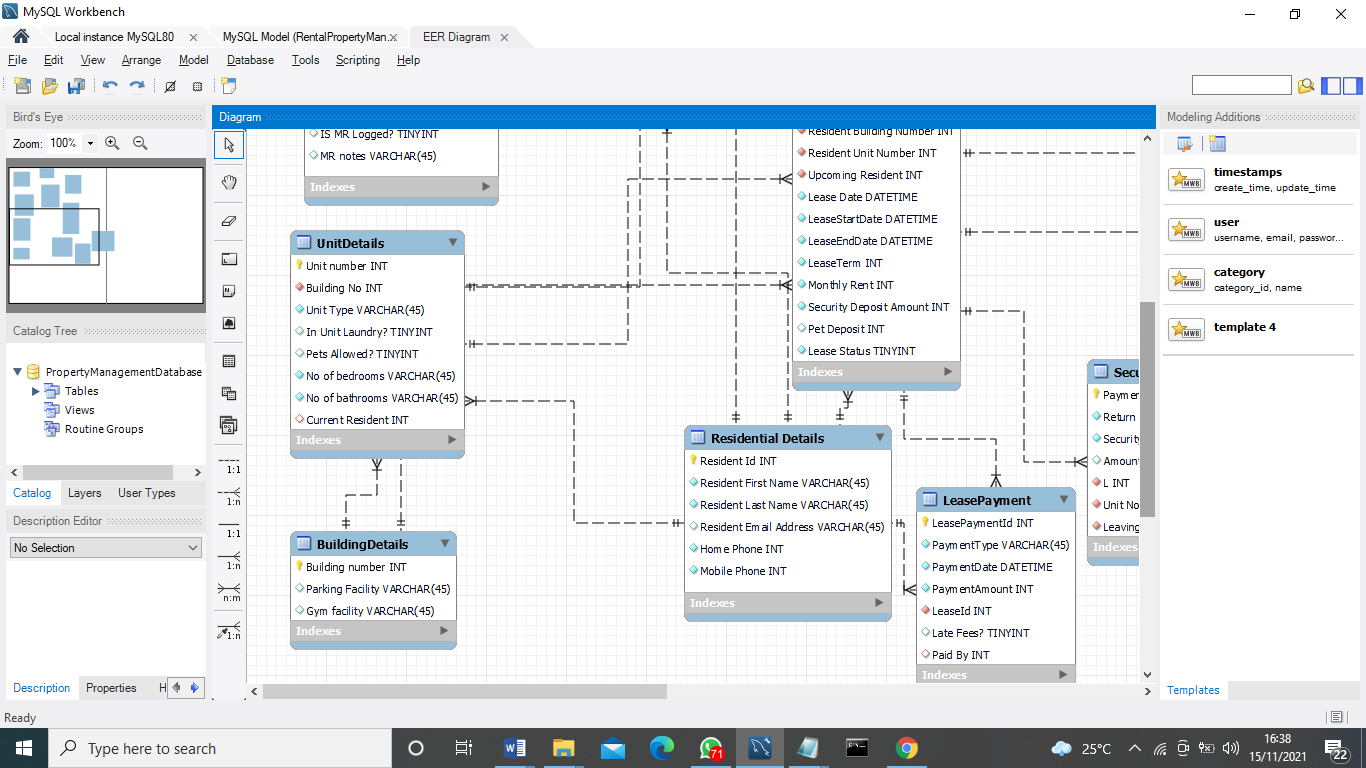
Rental management (house\_id, Tenant)

**Screenshots**

**Payment**







# Relational Database Design

*Description: This is where you will complete your database design. For each of the entity sets, you identified in the preceding section, analyze them to make sure they pass 2nd, 3rd, 4th, and Boyce-Codd Normal Form. If they do not, introduce additional entity sets or key changes to make sure that they do. Then, add foreign keys to connect entity sets that are related. For many-to-many relationships, introduce bridge entity sets to convert them into two one-to-many relationships. Also, consider whether you should introduce surrogate keys to create a more efficient primary key for some of your entity sets. Finally, diagram your design in Vertabello. Make sure your ER diagram correctly shows all entity sets, their primary and foreign keys, the data types for each attribute, and the connectivity and participation characteristics of each entity set. Your final Vertabello design should be something you could actually implement in a relational database management system.*

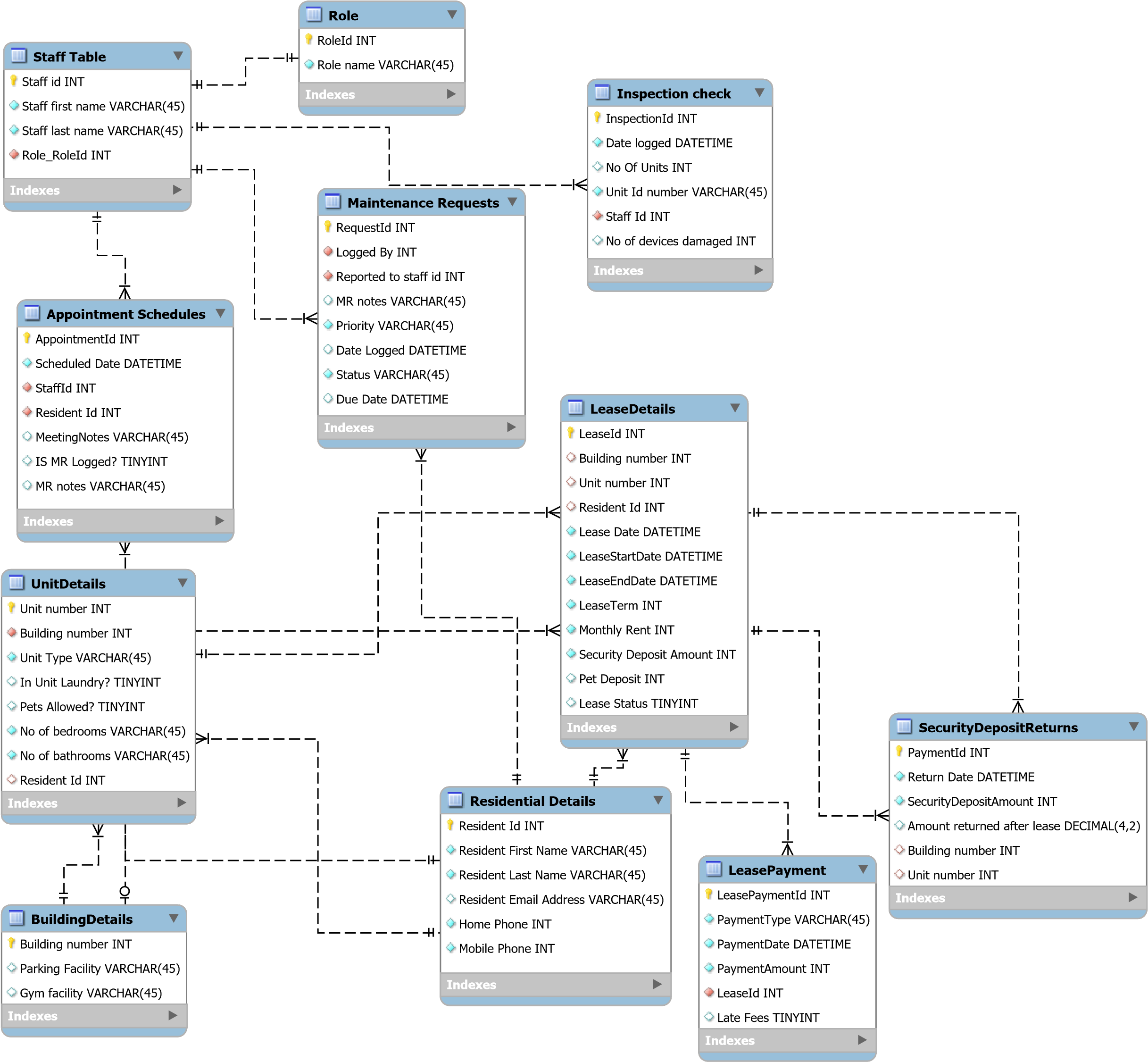
*Rubric: Your work will be graded as follows:*

* *4 points for the normalization analysis of your entity sets.*
* *3 points for introducing bridge entity sets.*
* *3 points for choosing foreign keys and perhaps more efficient surrogate keys*
* *10 points for correctly depicting your physical database model in Vertabello*

*Total points possible: 20*

ENTER YOUR RELATIONAL DATABASE DESIGN HERE

**Final Diagram**

****

# Data Definition Language (DDL) Scripts

*Description: Use Vertabello to generate a script of SQL commands that build the database and its table structures. Write scripts or build Excel spreadsheets that take your data files and generate scripts of SQL insert statements from them. Use the MySQL source command to run the various scripts needed to build and populate the database in MySQL. Include the source code and / or Excel spreadsheets you use to manipulate and populate the data. Make sure all your tables have at least three records in them and that you've linked the tables through their foreign keys.*

*Rubric: Your work will be graded as follows:*

* *Database and table creation statements from Vertabello saved as a sql script file: 3 points*
* *Scripts you write or Excel spreadsheets you create to generate SQL commands for populating the tables, uploaded to GitHub: 8 points*
* *Descriptions of the scripts and Excel spreadsheets you wrote along with code excerpts included in the Word document: 5 points*
* *Screenshots of your successful attempts to use the MySQL source command to populate each table with at least three records: 4 points*

*Total points possible: 20*

ENTER YOUR DDL WORK HERE

Defining data structures is the purpose of a computer language called the Data Definition Language (DDL). The Codasyl database model introduced the word about a Data Definition Language used to describe the user's data model in terms of records, fields, and "sets." However, the time has evolved to include any formal language used to describe data or information structures such as XML schemas.

**SQL statements**

**Populating tables**

-- inserting values in appointment schedule table --

INSERT INTO appointment\_schedules (Scheduled\_Date, StaffId, Resident\_Id, MeetingNotes, is\_MR\_Logged, MR\_notes)

values ('2021-11-13',1,2,'Payment queries solved','N','');

INSERT INTO appointment\_schedules (Scheduled\_Date, StaffId,Resident\_Id,MeetingNotes,is\_MR\_Logged,MR\_notes)

values('2021-11-13',3,1,'Maintenance Request','Y','Service the taps in washrooms.');

INSERT INTO appointment\_schedules(Scheduled\_Date,StaffId,Resident\_Id,MeetingNotes,is\_MR\_Logged,MR\_notes)

values('2021-11-13',2,2,'Feedback','N','');

INSERT INTO appointment\_schedules(Scheduled\_Date,StaffId,Resident\_Id,MeetingNotes,is\_MR\_Logged,MR\_notes)

values('2021-11-14',4,3,'Raised a Maintenance Request','Y','Water in sink doesnot drain');

INSERT INTO appointment\_schedules(Scheduled\_Date,StaffId,Resident\_Id,MeetingNotes,is\_MR\_Logged,MR\_notes)

values('2018-12-14',1,3,'','N','');

**Altering table scripts**

-- Reference: item\_leased\_unit (table: item\_leased)

ALTER TABLE `item\_leased` ADD CONSTRAINT `item\_leased\_unit` FOREIGN KEY `item\_leased\_unit` (`unit\_id`)

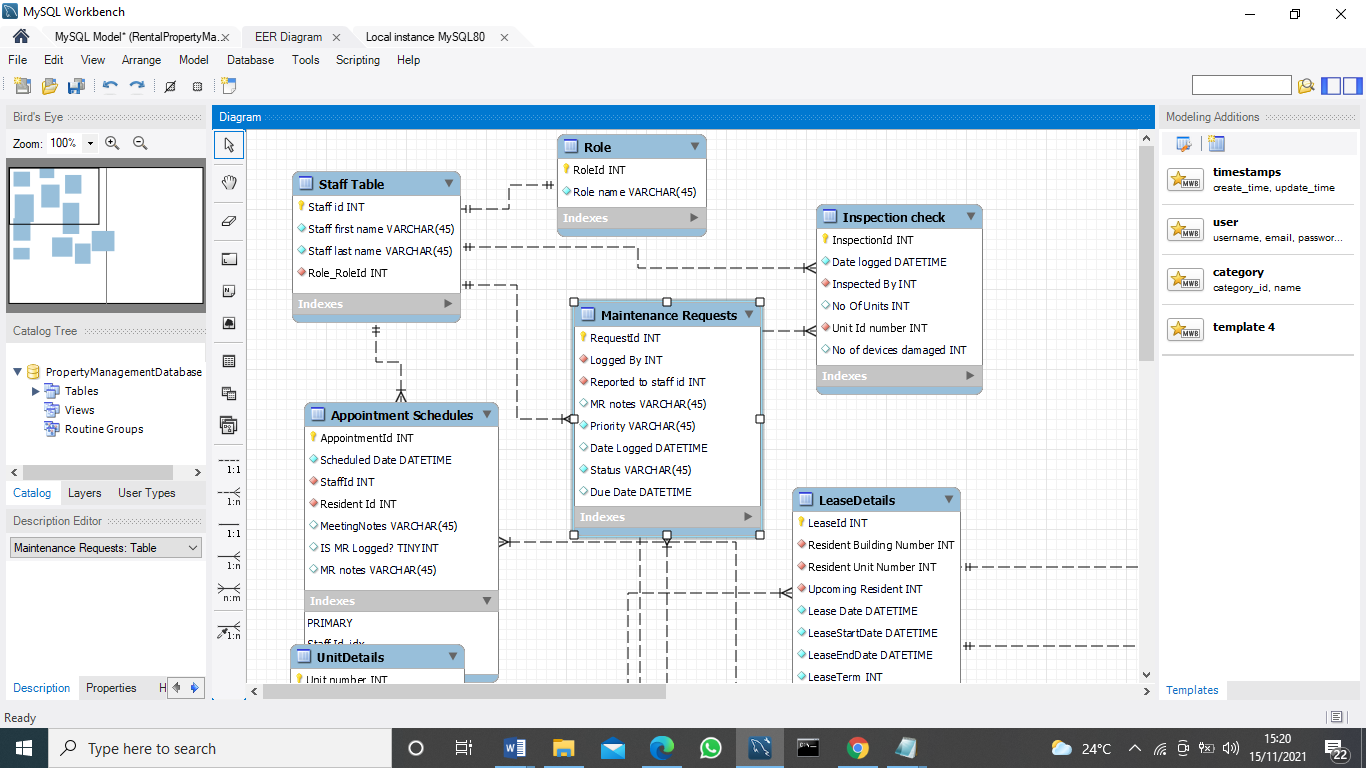
REFERENCES `unit` (`id`);

-- Reference: item\_leased\_user\_account (table: item\_leased)

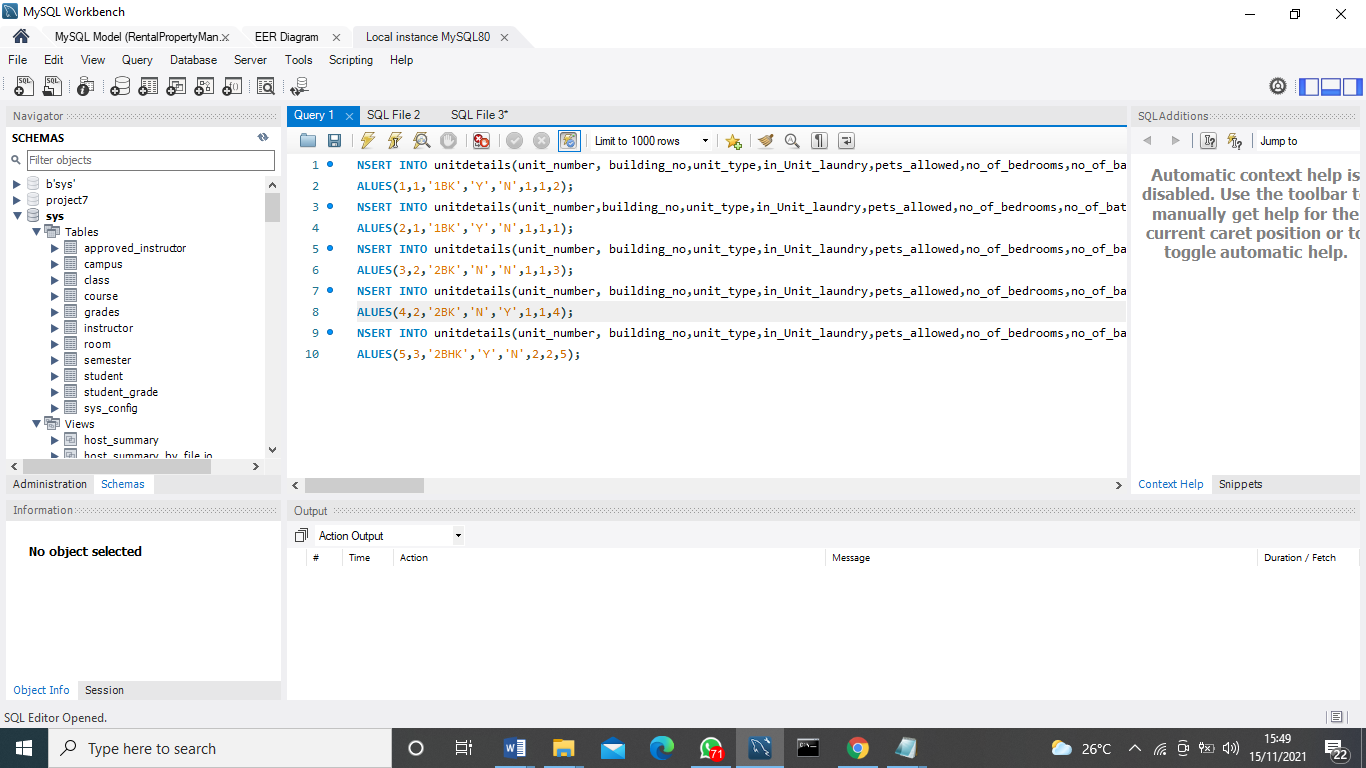
ALTER TABLE `item\_leased` ADD CONSTRAINT `item\_leased\_user\_account` FOREIGN KEY `item\_leased\_user\_account` (`renter\_id`)

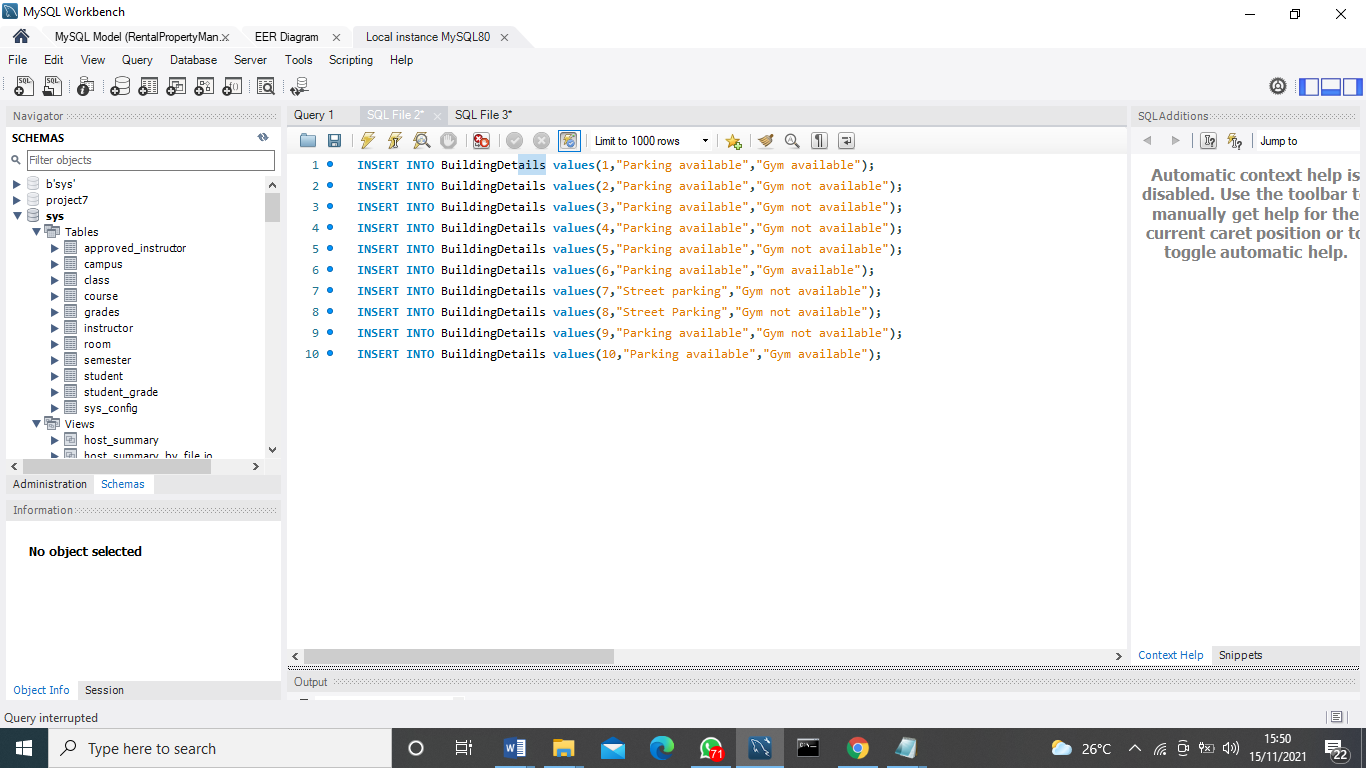
REFERENCES `user\_account` (`id`);

**Screenshots**



**Insert**





Data Manipulation Language Scripts

*Description: Write the SQL commands for twelve queries.* *Two queries should be inserted statements, two should update statements, one should be a delete statement, one should be a simple select statement that selects a subset of the rows and columns from one table, two should be a select statements that select data from a joining of two tables, two should use summary functions to generate statistics about the data, one should be a multi-table query, and one should be another query of your choice. Show the queries and screenshots of the results in your Word document, and save your queries in a commented sql script to GitHub.*

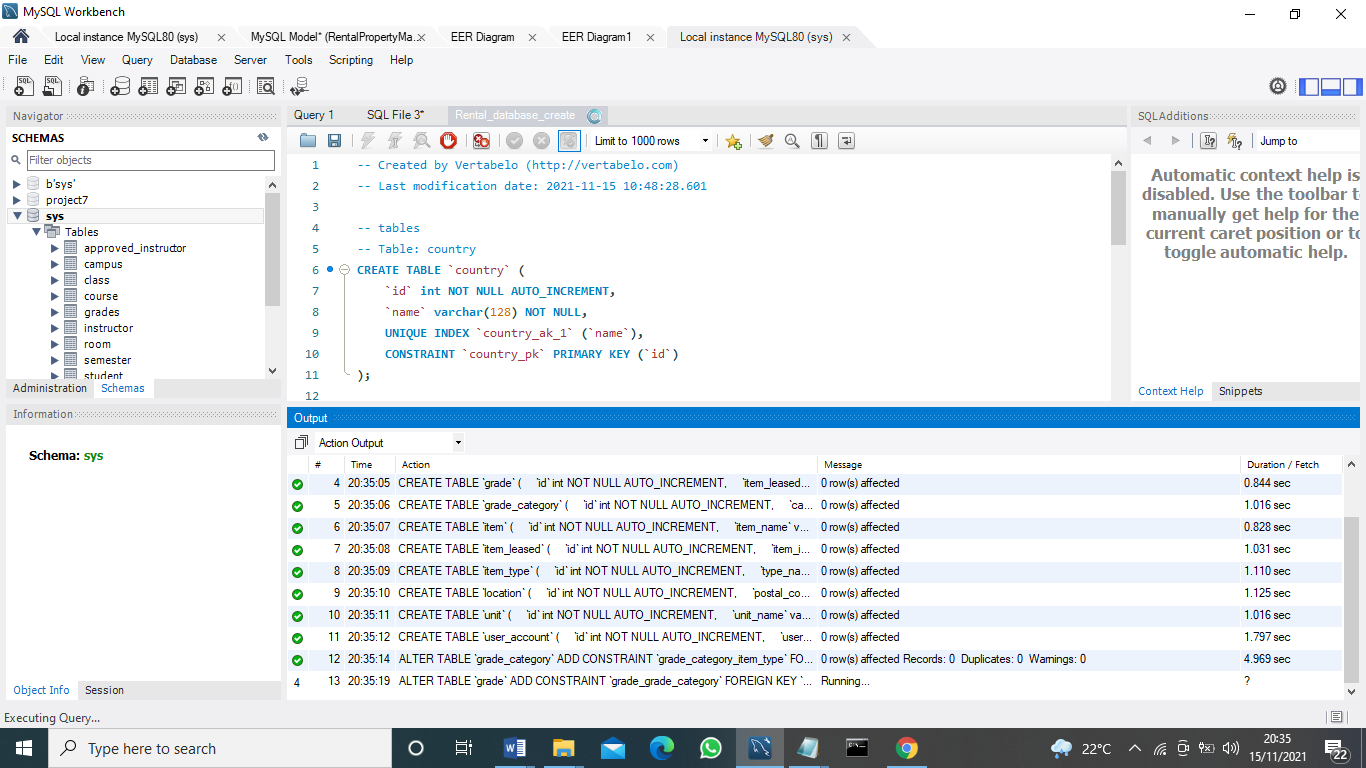
*Rubric: Your work will be graded as follows:*

* *1 point each for the two insert statements*
* *1 point each for the two update statements*
* *1 point for the delete statement*
* *1 point for the simple select statement*
* *2 points each for the 2 join statements*
* *2 points each for the two that use summary statements*
* *2 points for the multi-table query*
* *2 points for the query of your choice.*
* *12 points for showing the query and a screenshot of the corresponding result set back-to-back for each of these queries in your Word document.*

*Total points possible: 30*

ENTER DML WORK HERE

**SQL Script running in MySQL Workbench**



**Two insert statements**

INSERT INTO item\_leased(id,item\_id ,rent\_id ,price )

values(‘12',’house’,’22’ ,’10000’);

**B**

INSERT INTO item\_leased(id,item\_id ,rent\_id ,price )

values(‘13',’house’,’23’ ,’10000’);

**two update statements**

UPDATE item

SET items\_name = 'beinghun'

WHERE item\_Id = '3'

**B**

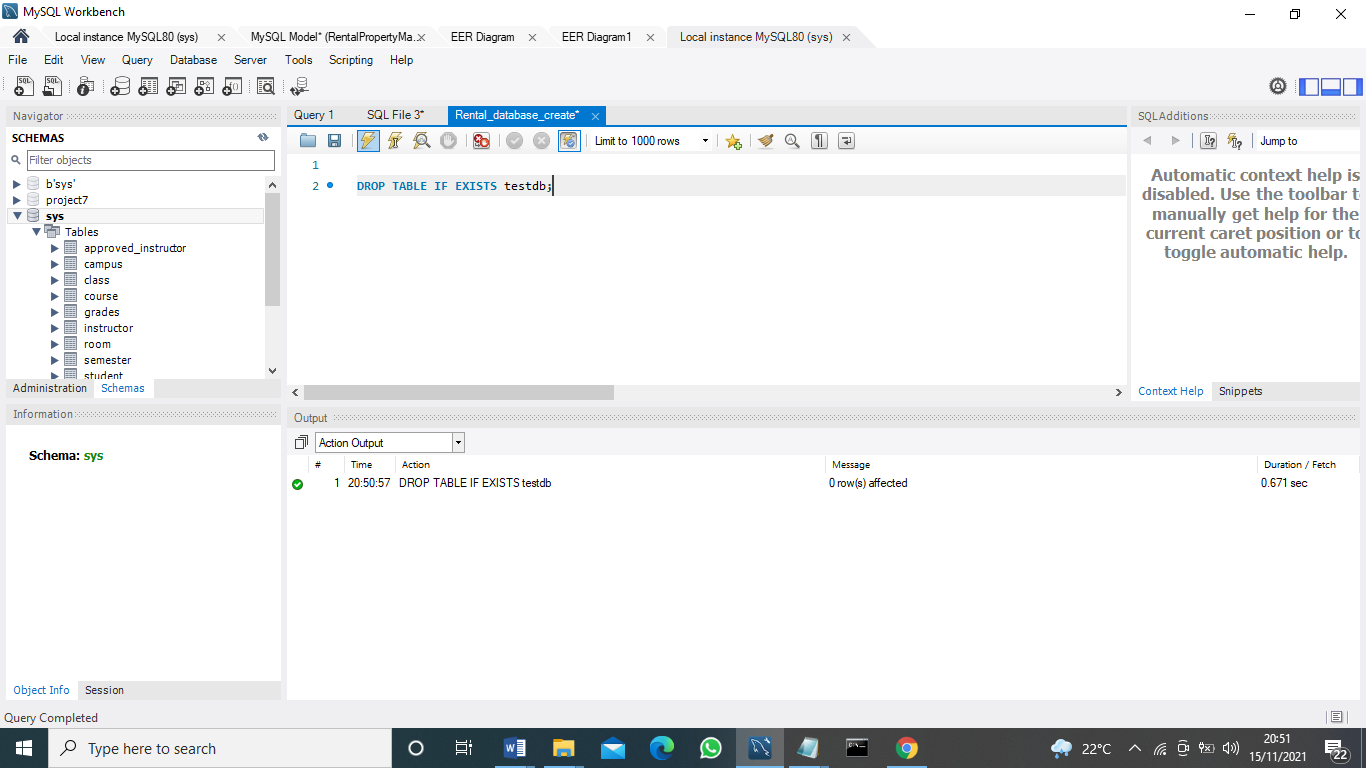
UPDATE item\_leased

SET items\_name = 'codein'

WHERE rent\_Id = '4'

**delete statement**

DROP TABLE IF EXISTS testdb;



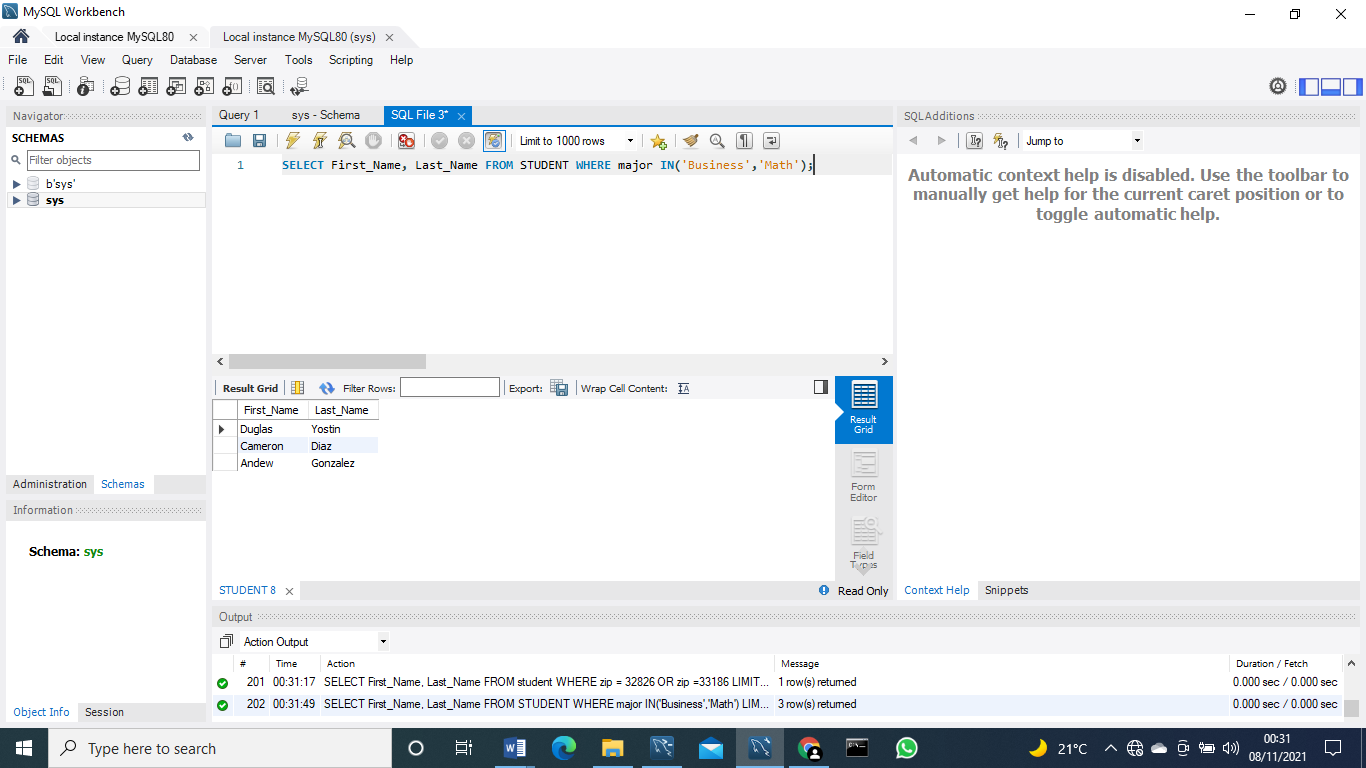
**simple select statement**

SELECT location\_id FROM item;

values('Ashm','Nigm','asd@gmail.com','9','90');

**for the 2 join statements**

SELECT id, item\_name, ‘item\_type\_id’, ‘location\_id’ item\_location  
FROM Orders  
INNER JOIN Item\_leased ON `id`, `item\_id`, `renter\_id`;



**2 points each for the two that use summary statements**

SELECT o.orderid, o.qty, i.itemprice, i.itemdesc

2FROM orders o

3LEFT JOIN items i

4on o.itemid = i.itemid

**Part b**

SELECT o.orderid, o.qty, i.itemprice, i.itemdesc

2FROM orders o

3LEFT JOIN items i

4on o.itemid = i.itemid

**2 points for the multi-table query**

SELECT id, item\_name

FROM item

INNER JOIN item\_leased

ON id =item\_id

ORDER BY id;

**Part 2**

SELECT id, item\_name

FROM item

INNER JOIN item\_leased

ON id =item\_id

ORDER BY id;

**2 points for the query**.

SELECT DISTINCT location FROM item;

B

SELECT DISTINCT location FROM item\_leased;

# Indexes

*Description: Improve the performance of your design by adding indexes to various tables. Show the SQL needed to add the indexes. Explain why you chose the ones you added. Explain how you would demonstrate the impact the indexes had on the performance of various queries.*

*Rubric: Your work will be graded as follows:*

* *6 points for clearly defining at least three indexes and explaining why you chose them.*
* *3 points for showing the sql needed to generate the indexes*
* *3 points for explaining how you would demonstrate the performance improvement afforded by the indexes.*

*Total points possible: 12*

ENTER YOUR INDEX WORK HERE

The main reasons for using the below view include:

Find rows that meet a WHERE clause rapidly.

In order to remove rows from the equation.

It's possible to utilize the leftmost prefix of a multiple-column index to find rows in a table and lastly, the perform joins to collect rows from other tables

INDEX `Resident Id\_idx` (`Upcoming Resident` ASC) VISIBLE,

INDEX `Building number\_idx` (`Resident Building Number` ASC) VISIBLE,

INDEX `LeaseId\_idx` (`LeaseId` ASC) VISIBLE,

**To what extent are indexes beneficial?**

One of the greatest methods to increase database application speed is to use effective indexes. In the absence of an index, the SQL Server engine is like a bookworm looking for a word in a book by paging through the whole text. It takes significantly less time to finish a job when utilizing the book's index at the back.

# Views

*Description: Add two views to your database to provide easy access to combinations of data from multiple tables.*

*Rubric: Your work will be graded as follows:*

* *2 points for including the SQL for generating the two views in your Word document*
* *2 points for including screenshots for the data contained in each view in your Word document*
* *2 points for explaining why each view is a valuable addition to your database*
* *2 points for explaining who might benefit most from having access to each view.*

*Total points possible: 8*

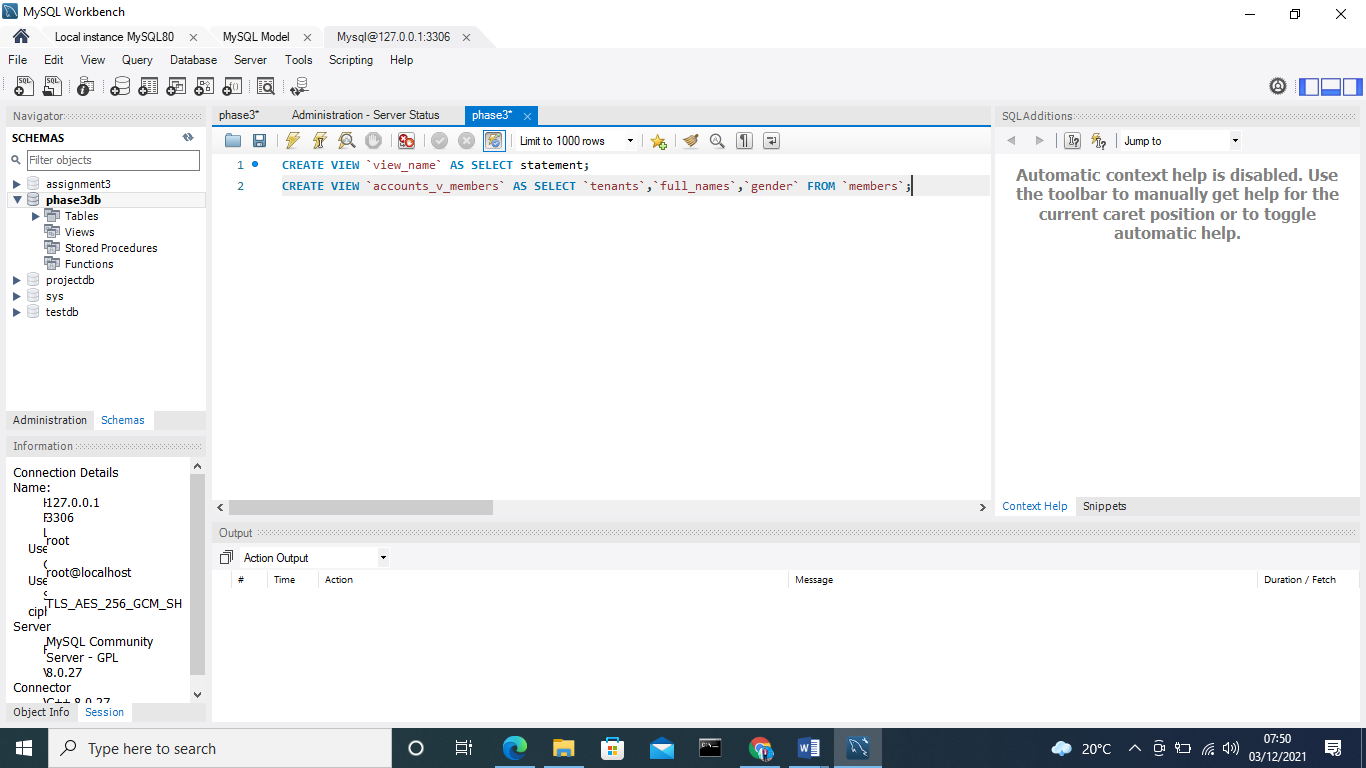
ENTER YOUR WORK WITH VIEWS HERE

Generating views

CREATE VIEW `view\_name` AS SELECT statement;

CREATE VIEW `accounts\_v\_members` AS SELECT `tenants`,`full\_names`,`gender` FROM `members`;

Screenshot



VIEWS are nothing more than virtual tables that show data from other tables instead of storing any data of their own. VIEWS are nothing more than SQL queries. You may have a view that includes all of the data in your database or only some of it. It is possible for a MySQL view to present data from one or more tables.

Only the rental department's allowed columns have been returned. In the member's table, more information has been omitted.

VIEWS enhance re-usability. In the future, you won't have to write complicated queries, including joins. VIEWS is used to reduce all the complexity into a single query. Condensed code may be easily integrated into your program, making it simpler to maintain. This will reduce the likelihood of mistakes and make your code easier to understand.

# Triggers

*Description: Add a trigger to a table so that data will be updated when a certain event occurs*

*Rubric: Your work will be graded as follows:*

* *2 points for including the SQL for the trigger in your Word document*
* *2 points for clearly explaining the purpose of the trigger*
* *2 points for a screenshot and explanation that shows the trigger in action.*

*Total points possible: 6*

ENTER YOUR WORK WITH TRIGGERS HERE

Create Trigers

CREATE TRIGGER before\_tenanats\_update

BEFORE UPDATE ON tenants

FOR EACH ROW

INSERT INTO tenanats\_audit

SET action = 'update',

tenanatsNumber = OLD.tenanatsNumber,

lastname = OLD.lastname,

changedat = NOW();

Trigger 2

CREATE TABLE tenanats (

id INT AUTO\_INCREMENT PRIMARY KEY,

tenanatsNumber INT NOT NULL,

lastname VARCHAR(50) NOT NULL,

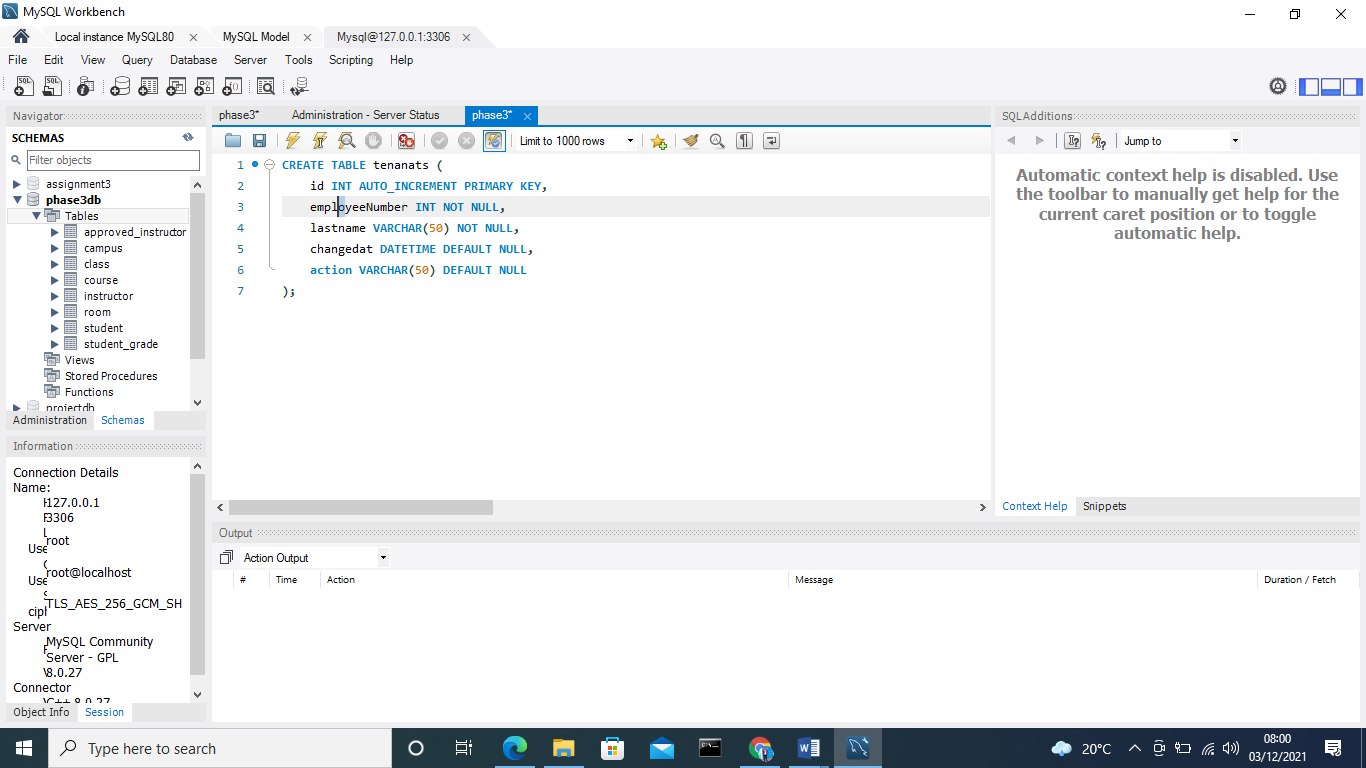
changedat DATETIME DEFAULT NULL,

action VARCHAR(50) DEFAULT NULL

);

A trigger is a stored program that is automatically launched in reaction to an event in the related table, such as an insert, update, or deletion.. Triggers may be defined that are automatically triggered when a new row is added to an existing table.

Screenshot



# Transactions

*Description: Demonstrate that you know how to define and use a transaction. Why are transactions important for ensuring ACID behavior?*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly explaining the importance of transactions to ensuring ACID behavior*
* *3 points for including a screenshot and accompanying explanation of a MySQL transaction.*

*Total points possible: 6*

ENTER YOUR WORK WITH TRANSACTIONS HERE

Transaction logs for the MySQL database

In order to keep track of transactions, MySQL offers the following statement:

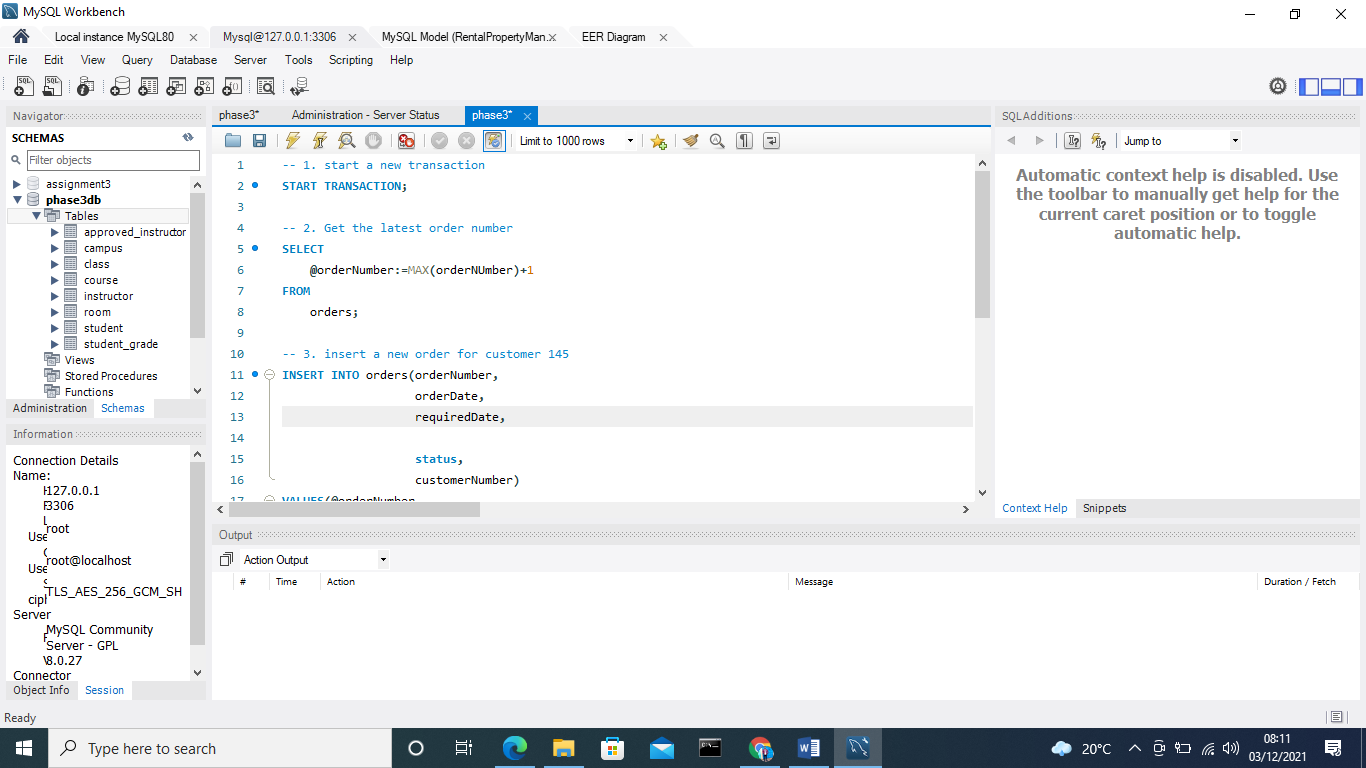
The START TRANSACTION statement is used to initiate a new transaction. The START TRANSACTION may also be referred to as the BEGIN or the BEGIN WORK command.

I am using the COMMIT statement to finalize and save the current transaction.

The ROLLBACK statement is used to revert the current transaction's modifications.

When you want to deactivate or enable the current transaction's auto-commit mode, use the SET auto-commit statement.

Screenshot



# Database Security

*Description: Identify the different kinds of users who will use your database. Write GRANT statements to define the privileges for these different kinds of users.*

*Rubric: Your work will be graded as follows:*

* *6 points for clearly identifying and describing the various kinds of users who will use the databases and identifying and justifying what privileges each should have.*
* *4 points for writing GRANT statements that assign privileges to these different kinds of users.*
* *4 points for demonstrating with screenshots that your GRANT statements do distinguish among different kinds of users in regard to what they can do with the database.*

*Total points possible: 14*

ENTER YOUR WORK WITH DATABASE SECURITY HERE

The system will need a Database Administrator (DBA).

If the user needs to access the database again, the DBA will establish a new user ID and password for them.

Users that have little DBMS expertise yet routinely utilize database programs in their everyday lives to get desired outcomes are known as parametric end users.

The System Analyst is a user who will examine the needs of parametric end users. They make sure that the needs of the end-users are met.

# Locking and Concurrent Access

*Description: Explain the purpose of locking tables and show how to do that to prevent inconsistencies that may arise in your data when concurrent transactions take place.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly explaining an example that shows why you should lock tables to prevent inconsistencies.*
* *3 points for providing a screenshot and accompanying explanation of locking tables.*

*Total points possible: 6*

ENTER YOUR WORK WITH LOCKING AND CONCURRENT ACCESS HERE

MySQL must ultimately commit any change you make to a table to disk, no matter how little the change is.

The process of making a change to a table with indexes is made considerably more difficult by the fact that MySQL must conduct the extra step of updating the indexes. When you make several changes to the data at the same time, such as by conducting an UPDATE to important (i.e., code) fields, the impact of this extra step might be magnified.

Depending on the circumstances, MySQL may be required to do hundreds or even thousands of write operations.

Locking tables

BEGIN;

SELECT @A:=SUM(rent) FROM tenanats WHERE type=1;

UPDATE transactions SET summmary=@A WHERE type=1;

COMMIT;

Command2

LOCK TABLES trans READ, customer WRITE;

Command 3

CREATE TRIGGER t1\_a\_ AFTER INSERT ON t1 FOR EACH ROW

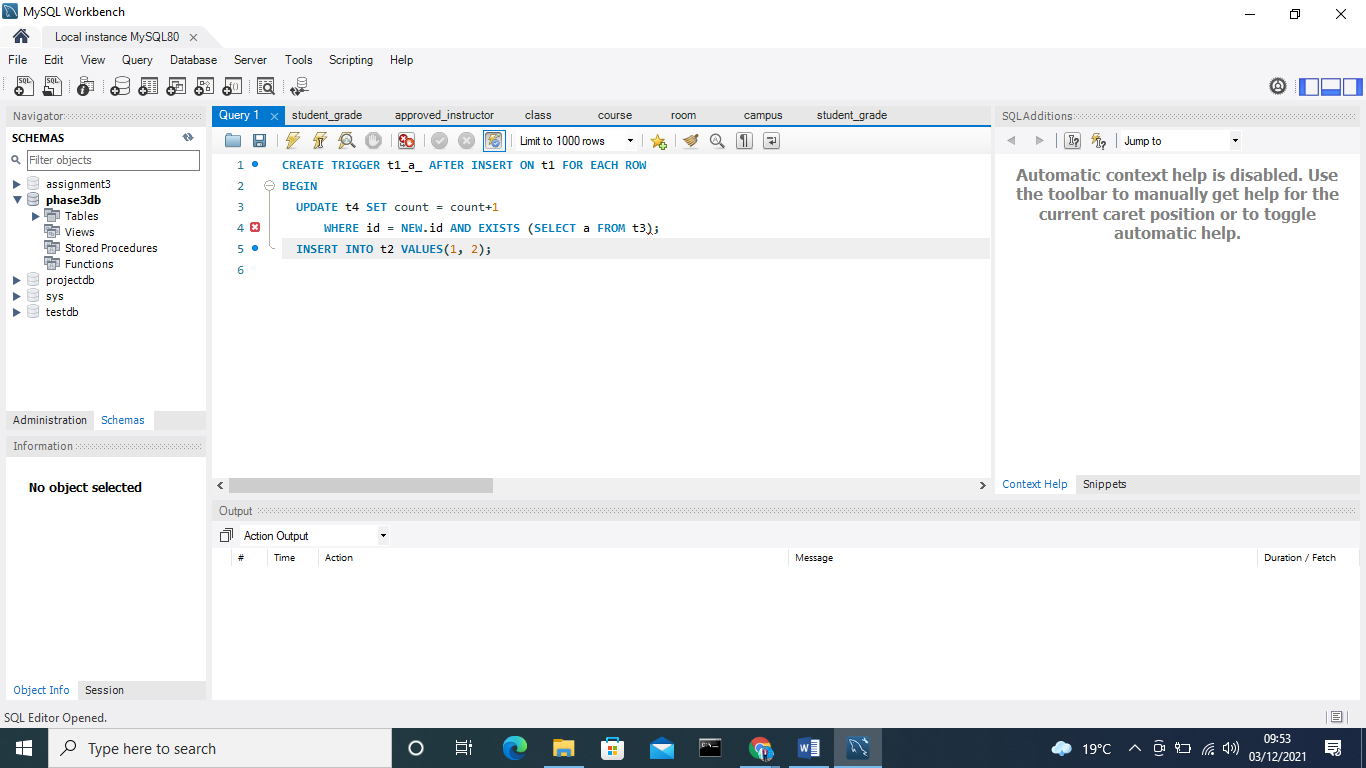
BEGIN

UPDATE t4 SET count = count+1

WHERE id = NEW.id AND EXISTS (SELECT a FROM t3);

INSERT INTO t2 VALUES(1, 2);

END;



# Backing Up Your Database

*Description: How you will back up your database. What commands will you issue? How frequently will the commands run? How can they be automated? Where will the backups be stored?*

*Rubric: Your work will be graded as follows:*

* *12 points for clearly explaining and justifying your database backup strategy, including the frequency with which you will back up the database, how you will automate backups, where you will store them, and how you will secure them. You will earn three points for addressing each factor (frequency, location, automation, and security)*
* *3 points for providing a screenshot of the command you would issue to back up the database and for including a portion of the resulting file.*

*Total points possible: 15*

ENTER YOUR WORK ON DATABASE BACKUPS HERE

# Python Programming

*Description: Write a Python program that generates a report that contains a subset of the data from your database. Include the code for your Python program in your Word document, and also post the program to your GitHub repository.*

*Rubric: Your work will be graded as follows:*

* *12 points for writing a Python script (and including its code in the Word doc) that will pull data from a database and store it to a text file and present it to the screen. Your code must have comments in it that explain how it works. You will be awarded 3 points for successfully connecting to the database, 3 points for successfully querying it, and 4 points for presenting the data to the screen and to a file. Internal comments count for 2 points.*
* *2 points for posting the code to GitHub*
* *4 points for showing a screenshot of your running the script and showing the results it produces on the screen.*

*Total points possible: 18*

ENTER YOUR PYTHON DATABASE PROGRAMMING WORK HERE

import pymysql

#import app

from app import app

#import mysql

from db import mysql

from flask import Flask, Response, render\_template

#route import

@app.route('/')

def upload\_form():

return render\_template('download.html')

@app.route('/download/report/pdf')

def download\_report():

conn = None

cursor = None

try:

conn = mysql.connect()

#connect to mysql

cursor = conn.cursor(pymysql.cursors.DictCursor)

cursor.execute("SELECT emp\_id, emp\_first\_name, emp\_last\_name, emp\_designation FROM employee")

result = cursor.fetchall()

#generating pdf

pdf = FPDF()

pdf.add\_page()

page\_width = pdf.w - 2 \* pdf.l\_margin

pdf.set\_font('Times','B',14.0)

pdf.cell(page\_width, 0.0, 'rental Data', align='C')

pdf.ln(10)

pdf.set\_font('Courier', '', 12)

col\_width = page\_width/4

pdf.ln(1)

th = pdf.font\_size

for row in result:

pdf.cell(col\_width, th, str(row['emp\_id']), border=1)

pdf.cell(col\_width, th, row['emp\_first\_name'], border=1)

pdf.cell(col\_width, th, row['emp\_last\_name'], border=1)

pdf.cell(col\_width, th, row['emp\_designation'], border=1)

pdf.ln(th)

pdf.ln(10)

pdf.set\_font('Times','',10.0)

pdf.cell(page\_width, 0.0, '- end of report -', align='C')

return Response(pdf.output(dest='S').encode('latin-1'), mimetype='application/pdf', headers={'Content-Disposition':'attachment;filename=employee\_report.pdf'})

except Exception as e:

print(e)

finally:

cursor.close()

conn.close()

if \_\_name\_\_ == "\_\_main\_\_":

app.run()

# PHP Programming

*Description: Build an HTML form that enables the user to specify criteria to search by. Use PHP to show the results of the query on a resulting web page. Make sure you include protections against an SQL injection attack. Include your HTML and PHP code in your Word document, and also post the files to your GitHub repository.*

*Rubric: Your work will be graded as follows:*

* *4 points for writing an HTML form the user will use to enter search criteria*
* *8 points for a PHP script that uses the search criteria and returns results*
* *4 points for an HTML page that shows the results*
* *4 points for explaining what SQL injection might be run on your website and explaining how you prevented it.*
* *4 points for providing screen shots of your PHP website in action.*
* *2 points for posting your code to GitHub*

*Total points possible: 26*

ENTER YOUR PHP DATABASE APP PROGRAMMING WORK HERE

***HTML Form***

<head> <title>Search</title> <meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> <link rel="stylesheet" type="text/css" href="style.css"/></head><body> <form action="search.php" method="GET"> <input type="text" name="query" /> <input type="submit" value="Search" /> </form></body></html>

***PHP script that uses the search***

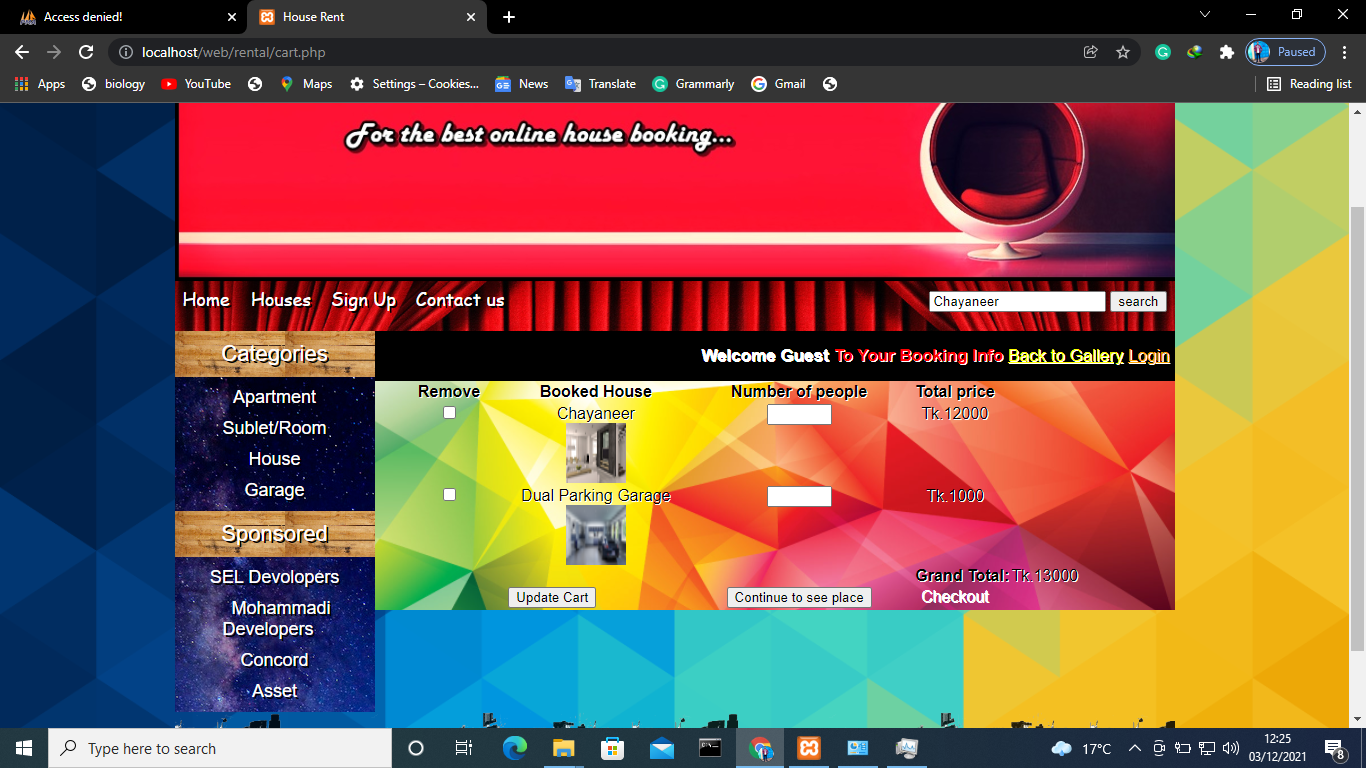
<?php $query = $\_GET['query']; $min\_length = 3 if(strlen($query) >= $min\_length){ $query = htmlspecialchars($query); $query = mysql\_real\_escape\_string($query); // no SQL injection $raw\_results = mysql\_query("SELECT \* FROM articles WHERE (`title` LIKE '%".$query."%') OR (`text` LIKE '%".$query."%')") or die(mysql\_error());

if(mysql\_num\_rows($raw\_results) > 0){ while($results = mysql\_fetch\_array($raw\_results)){ echo "<p><h3>".$results['title']."</h3>".$results['text']."</p>"; } else{ echo "No results"; } } else{ echo "Minimum length is ".$min\_length; }?>

**Search Function**



Running web



**SQL injections**

**In-band SQL**

An in-band SQL attack is one where the attacker utilizes the same communication channel for both the attack and the collection of the results. Because of its ease of use and effectiveness, in-band SQL attacks are a widespread occurrence.

**Blind SQL**

As the server receives data payloads and responds, an attacker might learn more about its structure by observing its reaction and behavior. Blind SQL is a term for a technique in which the data is not sent from the website database to the attacker, preventing the attacker from seeing any information about the attack in real time.

**Out of band SQL Injection**

The web application's database server must have specific functionalities enabled in order for this kind of attack to work. SQL injection attacks are often used as an alternative to the in-band and inferential SQLi methods.

# Suggested Future Work

*Description: Describe the limitations of your current database and explain how you or someone else could improve the design to address these shortcomings. Also describe how you might take advantage of leverage cloud services to increase the performance and availability of your database. Finally, explain the advantages and disadvantages of storing your data in a NoSQL format instead.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly describing the limitations of your databases*
* *3 points for explaining how you would address these shortcomings*
* *3 points for explaining how you might migrate the database to the cloud and describing what advantages you might gain from doing that.*
* *3 points for explaining the advantages and disadvantages of storing your data in a document-based NoSQL format instead.*

*Total points possible: 12*

ENTER YOUR SUGGESTED FUTURE WORK IDEAS HERE

**Database errors**

**Error in the database**

A database contains all of the information that is important to rental. As a result, the database must operate at peak performance and with no malfunctions. A database failure may be very damaging since it can result in the loss or corruption of extremely sensitive information.

**High Hardware Expenses**

A database holds a massive quantity of information. As a result, a considerable disk capacity is necessary to store all of this information. More storage space may be required. All of this raises the cost of hardware significantly, making a database prohibitively costly.

**Expenses Associated with Data Conversion**

This cost may potentially be more than the price of database construction and administration in certain cases. This is one of the reasons why most businesses choose to continue working on their existing databases rather than upgrading to new ones.

**Control to database limitations**

**Performance**

Databases have always had to meet stringent performance requirements. Although the quantity of data you are collecting is increasing, pleasing your consumers, you must process it more quickly without having them go through unnecessary hoops to get there.

**Data security is really important.**

Data must be processed so that nothing is lost throughout the process. It is necessary to provide ACID guarantees throughout the whole database and the entire cluster, even whether the database is a non-relational or distributed cluster.

**Optimal use of available resources**

It used to be expected that databases would operate on the finest hardware available and act according to this assumption. That is no longer the case, thanks to the widespread use of containers and virtual machine deployment.

Advantages of migrating data to the cloud

Data Transfer to the Cloud has various advantages, and many businesses have already reaped the benefits of data migration to the cloud. However, here are a few of the most important advantages of data migration:

**Save a Bunch on Data Storage by Using Cloud Storage Services**

Businesses may avoid making large, up-front expenditures on servers, PCs, and other data storage equipment by moving data to the cloud. Because of this, organizations may store massive amounts of data in the cloud without investing in expensive IT infrastructure.

**Expanded Collaboration and Easier Scalability.**

When companies need to work together on a project, moving their data to the cloud may be beneficial. Such personnel have access to cloud-based data and may collaborate on papers simultaneously. Because the data is stored in the cloud, team members may access it at any time and from any location.

**A dependable backup option**

It is common for busy enterprises to overlook the need to regularly back up all their data on local servers or PCs. Data transfer to the cloud, on the other hand, provides a strong backup strategy. As a result of the superiority of cloud providers' data backup solutions and cloud data management software, the data backup is regularly carried out to avoid data loss.

**Adavanatages of NOSQL**

Advantages

Schema-less. Data may be stored in any format and any arrangement. If you're dealing with a constantly evolving system, this is a great way to keep track of enormous amounts of data.

You have improved speed and quality. Once you've created the document, the only thing left to do is to add your complicated object.

Only one foreign key. Papers can exist in isolation from one another without this relational dynamic.

Formats that are free to use. XML, JSON, and other variants to describe documents in a clean build process.

A built-in versioning system is included. Your papers can get more complicated as they expand in size. It reduces the likelihood of a fight.

**Disadvantages**

Atomicity weaknesses. JOINs aren't required to make changes to data in a relational database. It's possible to update all of your data with a single command.

Security. Almost 50% of all online apps are now leaking private information. For those who use NoSQL databases, web app vulnerabilities are a concern.

Activity Log

*Description: As an appendix, the team will keep a daily diary or log of their activity. What did you or your team study in this class each day? What did you learn? What did you accomplish or build or design? You don't have to enter something every day, but there should be at least three entries each week. Since we have eight weeks, that means you should make 3 posts to the Activity Log each week, for a total of at least 24 posts. Each post will be worth 1 point.*

*If you are working as part of a team, make sure you clearly identify which team member worked on which tasks. The Activity Log should help me figure out how each team member contributed to the project. If I cannot discern who worked on what aspects of the project from the activity log, no points will be awarded for it.*

*Total points possible: 24*

MAKE AT LEAST THREE ENTRIES PER WEEK. CLEARLY IDENTIFY WHAT EACH PERSON ON YOUR TEAM ACCOMPLISHED. YOU MUST SHARE THE RESPONSIBILITY OF COMPLETING THE PROJECT.