

# MIS686 Final Project - Peter Dudziak

## Table of Contents

- Part 1: Topic Selection
  - Selected Topic - [Parking Garage Management Software]
  - Overview
  - Business Rules
  - Use Cases
- Part 2: Database Design
  - ERD
  - Relational Diagram
- Part 3: Database Implementation
  - SQL Statements
    - Stored Procedures, Triggers, & Indexes
  - Data Implementation
- Part 4: Database Deployment
  - Database Deployment Screenshots
- Part 5: Questions & Dashboard
  - Analytical Questions
  - Dashboard Images

## Part 2: Topic Selection

**Selected Topic:** Parking Garage Management Software

### Overview:

The database that was created was made with the intent to track information related to a parking garage and the customers that frequented it. It allows the parking garage to track customers that come into the parking garage, find which parking spots they use, the time they spent in the parking spots, and the amount that they need to pay for parking in the parking garage for that long. It also keeps track of the customer's information in relation to their vehicle, including their license plate and car model. Customers can make reservations for certain parking spots before they arrive and access the garage, and the parking garage also has opening and closing hours that dictate when customers can enter and exit. This data is intended to give an idea of how frequently the parking garage is used, and how much of a profit it makes, while also committing the information of customers that use it for user analytics.

### Business Rules:

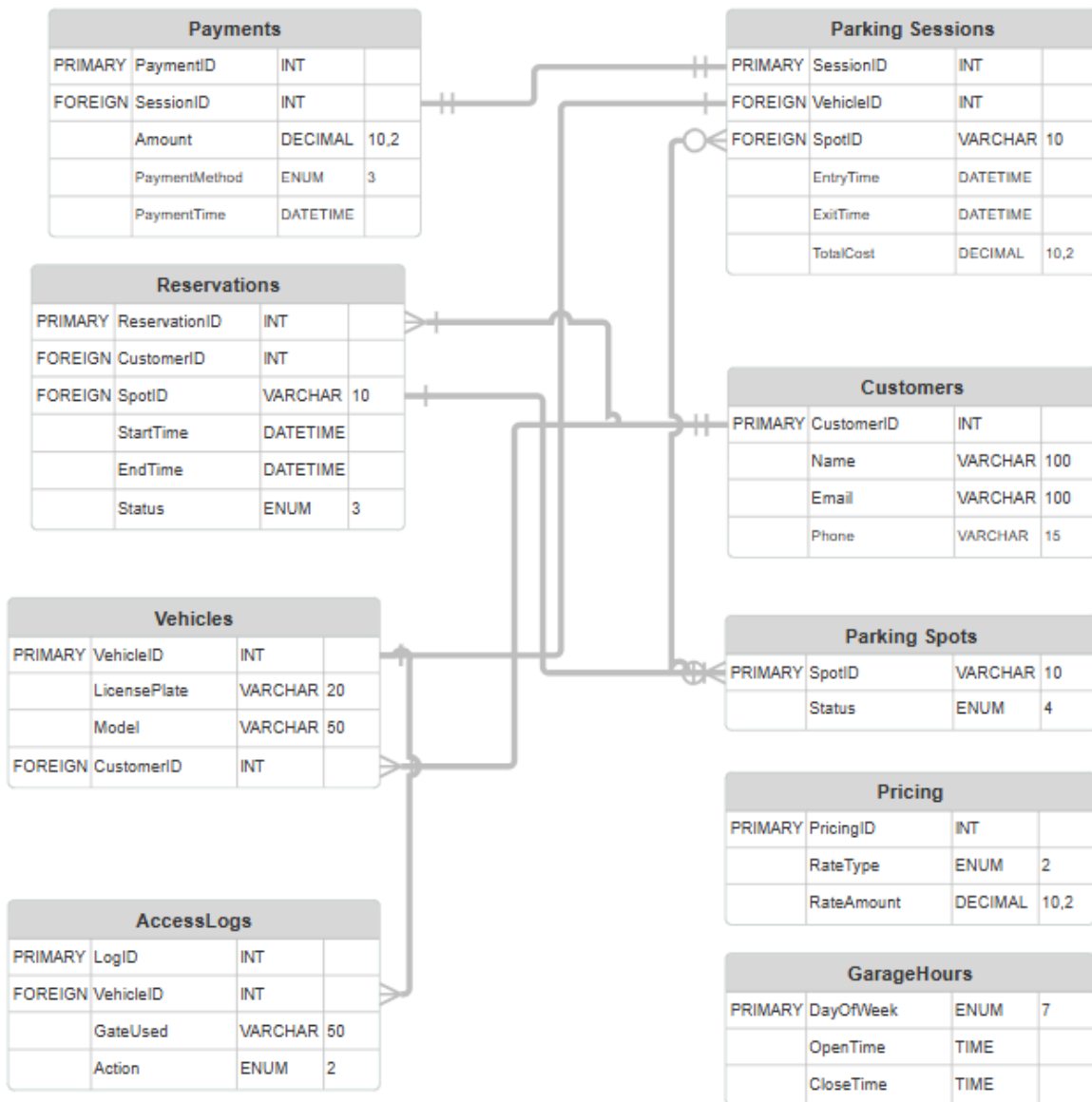
1. Each customer must have a car.
2. That car must have a valid license plate.
3. The customer must park in only one parking spot for the duration of their session.
4. The customer can have multiple sessions occurring at the same time.
5. A parking spot can be used multiple times per day, so long as there is no overlap in the parking sessions for that parking spot.
6. The customer cannot enter the parking garage outside of its normal working hours.
7. The customer will be charged an hourly fee depending on how long they stay in the parking garage for.
8. Each parking spot can only have one car at a time, and will be considered 'occupied' while in use. No other car can be parked in that space while it is occupied.
9. When a car leaves a parking space, it becomes vacant and available for use.
10. A parking space can be reserved or under maintenance. In which case, it cannot be parked in.
11. There is a 10.1% tax applied to the hourly parking fee, and a 2 dollar fee to reserve a parking space.

**Use Cases:**

1. Gathering data on how many people use the parking garage can help to promote the parking garage by setting up stronger fees. Finding information on when and how much certain people park in the garage can give insight into how to properly price the fees in the parking garage.
2. Tracking individual drivers by their license plate numbers and payment information can help in the event of a criminal investigation, wherein information regarding suspected persons is necessary for police handling.
3. Using historical data to make plans and determine when would be the best time for certain parking spaces to go under maintenance, or when to work on the parking garage's different sectors.

**Part 2: Database Design****ERD:**

The ERD (Entity Relationship Diagram) was created using the business rules to create a visible showcase of the different entities, attributes, and relationships to show how each table connects with one another.



### Relational Diagram:

From the ERD and expected relationships, it was transformed into a relational diagram, accounting for supertype relationships as well as weak entity relationships. Through an iterative process, primary keys, foreign keys, and relationships were created to develop a database schema of tables.



```
SpotID VARCHAR(10) PRIMARY KEY UNIQUE, -- e.g., A1, B2, Z30
Status ENUM('Available', 'Occupied', 'Maintenance') DEFAULT 'Available'
);
```

```
CREATE TABLE ParkingSessions (
    SessionID INT PRIMARY KEY AUTO_INCREMENT,
    VehicleID INT,
    SpotID VARCHAR(10),
    EntryTime DATETIME NOT NULL,
    ExitTime DATETIME,
    TotalCost DECIMAL(10, 2),
    FOREIGN KEY (VehicleID) REFERENCES Vehicles(VehicleID) ON DELETE CASCADE,
    FOREIGN KEY (SpotID) REFERENCES ParkingSpots(SpotID) ON DELETE CASCADE,
    INDEX (EntryTime),
    INDEX (ExitTime)
);
```

```
CREATE TABLE Payments (
    PaymentID INT PRIMARY KEY AUTO_INCREMENT,
    SessionID INT,
    Amount DECIMAL(10, 2) NOT NULL,
    PaymentMethod ENUM('Credit Card', 'Cash', 'Mobile Payment'),
    PaymentTime DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (SessionID) REFERENCES ParkingSessions(SessionID) ON DELETE
    CASCADE,
    INDEX (PaymentTime)
);
```

```
CREATE TABLE AccessLogs (
    LogID INT PRIMARY KEY AUTO_INCREMENT,
    VehicleID INT,
    GateUsed VARCHAR(50),
    Action ENUM('Entry', 'Exit'),
    Timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (VehicleID) REFERENCES Vehicles(VehicleID) ON DELETE CASCADE,
    INDEX (Timestamp)
);
```

```
CREATE TABLE Pricing (
    PricingID INT PRIMARY KEY AUTO_INCREMENT,
    RateType ENUM('Hourly', 'Overnight'),
    RateAmount DECIMAL(10, 2) NOT NULL,
    ValidFrom DATETIME NOT NULL,
```

```

ValidTo DATETIME,
INDEX (RateType)
);

CREATE TABLE GarageHours (
    DayOfWeek ENUM('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
'Sunday') PRIMARY KEY,
    OpenTime TIME NOT NULL,
    CloseTime TIME NOT NULL
);

CREATE TABLE Reservations (
    ReservationID INT PRIMARY KEY AUTO_INCREMENT,
    CustomerID INT,
    SpotID VARCHAR(10),
    StartTime DATETIME NOT NULL,
    EndTime DATETIME NOT NULL,
    Status ENUM('Active', 'Canceled', 'Completed') DEFAULT 'Active',
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) ON DELETE
CASCADE,
    FOREIGN KEY (SpotID) REFERENCES ParkingSpots(SpotID) ON DELETE CASCADE,
    INDEX (StartTime),
    INDEX (EndTime)
);

```

### **Dummy Data Generation:**

Dummy data is automatically generated using a random number generator, filling in the tables with a list of data and pulling from there.

```

INSERT INTO Customers (Name, Phone, Email)
WITH RECURSIVE NumberSequence AS (
    SELECT 1 AS num
    UNION ALL
    SELECT num + 1
    FROM NumberSequence
    WHERE num < 100
)
SELECT
    CONCAT('Customer', FLOOR(RAND() * 1000)), -- Random name
    CONCAT('555-', FLOOR(RAND() * 1000), '-', FLOOR(RAND() * 10000)),
    CONCAT('customer', FLOOR(RAND() * 1000), '_', num, '@example.com')

```

```

FROM NumberSequence;
CALL GenerateUniqueSpotIDs(100);
INSERT INTO Vehicles (LicensePlate, Model, CustomerID)
SELECT
    CONCAT(CHAR(65 + FLOOR(RAND() * 26)), CHAR(65 + FLOOR(RAND() * 26)),
    FLOOR(RAND() * 1000)),
    ELT(FLOOR(RAND() * 5) + 1, 'Toyota Camry', 'Honda Civic', 'Ford Mustang', 'Tesla Model S',
    'Chevrolet Tahoe'),
    FLOOR(RAND() * 100) + 1 FROM (SELECT 1 UNION SELECT 2 UNION SELECT 3 UNION
    SELECT 4 UNION SELECT 5 UNION SELECT 6 UNION SELECT 7 UNION SELECT 8 UNION
    SELECT 9 UNION SELECT 10) AS a
    CROSS JOIN (SELECT 1 UNION SELECT 2 UNION SELECT 3 UNION SELECT 4 UNION
    SELECT 5 UNION SELECT 6 UNION SELECT 7 UNION SELECT 8 UNION SELECT 9 UNION
    SELECT 10) AS b;

```

```

INSERT INTO ParkingSessions (VehicleID, SpotID, EntryTime, ExitTime, TotalCost)
SELECT
    v.VehicleID,
    ps.SpotID,
    NOW() - INTERVAL FLOOR(RAND() * 7) DAY - INTERVAL FLOOR(RAND() * 24) HOUR -
    INTERVAL FLOOR(RAND() * 60) MINUTE AS EntryTime,
    NOW() - INTERVAL FLOOR(RAND() * 7) DAY - INTERVAL FLOOR(RAND() * 24) HOUR -
    INTERVAL FLOOR(RAND() * 60) MINUTE + INTERVAL FLOOR(1 + RAND() * 24) HOUR AS
    ExitTime,
    ROUND((FLOOR(1 + RAND() * 24)) * 5.00, 2) AS TotalCost
FROM Vehicles v
JOIN ParkingSpots ps ON ps.Status = 'Available';

```

```

INSERT INTO Payments (SessionID, Amount, PaymentMethod)
SELECT
    ps.SessionID,
    ps.TotalCost + IF(RAND() < 0.5, 2, 0),
    ELT(FLOOR(RAND() * 3) + 1, 'Credit Card', 'Cash', 'Mobile Payment')
FROM ParkingSessions ps;

```

```

INSERT INTO AccessLogs (VehicleID, GateUsed, Action, Timestamp)
SELECT
    ps.VehicleID,
    CONCAT('Gate ', FLOOR(RAND() * 4) + 1),
    'Entry',
    ps.EntryTime
FROM ParkingSessions ps
UNION ALL

```

```

SELECT
    ps.VehicleID,
    CONCAT('Gate ', FLOOR(RAND() * 4) + 1),
    'Exit',
    ps.ExitTime
FROM ParkingSessions ps
WHERE ps.ExitTime IS NOT NULL;

```

```

INSERT INTO Pricing (RateType, RateAmount, ValidFrom, ValidTo) VALUES
('Hourly', 5.00, '2023-01-01 00:00:00', NULL);

```

```

INSERT INTO Reservations (CustomerID, SpotID, StartTime, EndTime, Status)
SELECT
    c.CustomerID,
    ps.SpotID,
    NOW() + INTERVAL FLOOR(RAND() * 30) DAY + INTERVAL FLOOR(RAND() * 24) HOUR,
    NOW() + INTERVAL FLOOR(RAND() * 30) DAY + INTERVAL FLOOR(RAND() * 24) HOUR +
INTERVAL FLOOR(RAND() * 24) HOUR,
    'Active'
FROM Customers c
JOIN ParkingSpots ps ON ps.Status = 'Available'
LIMIT 50;

```

### **Stored Procedures, Triggers, Views & Indexes:**

The majority of indexes used in this database are within the CREATE TABLE statements.

-- Indexes --

```

CREATE TABLE ParkingSessions (
    SessionID INT PRIMARY KEY AUTO_INCREMENT,
    VehicleID INT,
    SpotID VARCHAR(10),
    EntryTime DATETIME NOT NULL,
    ExitTime DATETIME,
    TotalCost DECIMAL(10, 2),
    FOREIGN KEY (VehicleID) REFERENCES Vehicles(VehicleID) ON DELETE CASCADE,
    FOREIGN KEY (SpotID) REFERENCES ParkingSpots(SpotID) ON DELETE CASCADE,
    INDEX (EntryTime),
    INDEX (ExitTime)
);

```



```
CREATE TABLE Payments (  
    PaymentID INT PRIMARY KEY AUTO_INCREMENT,  
    SessionID INT,  
    Amount DECIMAL(10, 2) NOT NULL,  
    PaymentMethod ENUM('Credit Card', 'Cash', 'Mobile Payment'),  
    PaymentTime DATETIME DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (SessionID) REFERENCES ParkingSessions(SessionID) ON DELETE  
CASCADE,  
    INDEX (PaymentTime)  
);
```

```
CREATE TABLE AccessLogs (  
    LogID INT PRIMARY KEY AUTO_INCREMENT,  
    VehicleID INT,  
    GateUsed VARCHAR(50),  
    Action ENUM('Entry', 'Exit'),  
    Timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (VehicleID) REFERENCES Vehicles(VehicleID) ON DELETE CASCADE,  
    INDEX (Timestamp)  
);
```

```
CREATE TABLE Pricing (  
    PricingID INT PRIMARY KEY AUTO_INCREMENT,  
    RateType ENUM('Hourly', 'Overnight'),  
    RateAmount DECIMAL(10, 2) NOT NULL,  
    ValidFrom DATETIME NOT NULL,  
    ValidTo DATETIME,  
    INDEX (RateType)  
);
```

```
CREATE TABLE Reservations (  
    ReservationID INT PRIMARY KEY AUTO_INCREMENT,  
    CustomerID INT,  
    SpotID VARCHAR(10),  
    StartTime DATETIME NOT NULL,  
    EndTime DATETIME NOT NULL,  
    Status ENUM('Active', 'Canceled', 'Completed') DEFAULT 'Active',  
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) ON DELETE  
CASCADE,  
    FOREIGN KEY (SpotID) REFERENCES ParkingSpots(SpotID) ON DELETE CASCADE,  
    INDEX (StartTime),  
    INDEX (EndTime)  
);
```

-- Triggers & Stored Procedures --

DELIMITER //

```
CREATE PROCEDURE GenerateUniqueSpotIDs(IN num_spots INT)
BEGIN
    DECLARE i INT DEFAULT 0;
    DECLARE spot_id VARCHAR(10);
    DECLARE letter CHAR(1);
    DECLARE number INT;
    WHILE i < num_spots DO
        SET letter = CHAR(65 + FLOOR(RAND() * 26));
        SET number = FLOOR(RAND() * 30) + 1;
        SET spot_id = CONCAT(letter, number);
        IF NOT EXISTS (SELECT 1 FROM ParkingSpots WHERE SpotID = spot_id) THEN
INSERT INTO ParkingSpots (SpotID, Status) VALUES (spot_id, 'Available');
            SET i = i + 1;
        END IF;
    END WHILE;
END //
```

```
CREATE PROCEDURE StartParkingSession(
    IN p_VehicleID INT,
    IN p_SpotID VARCHAR(10),
    IN p_EntryTime DATETIME
)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        RESIGNAL;
    END;
    START TRANSACTION;
    INSERT INTO ParkingSessions (VehicleID, SpotID, EntryTime)
VALUES (p_VehicleID, p_SpotID, p_EntryTime);
    UPDATE ParkingSpots
    SET Status = 'Occupied'
    WHERE SpotID = p_SpotID;
    COMMIT;
END //
```

```
CREATE PROCEDURE EndParkingSession(
```

```

    IN p_SessionID INT,
    IN p_ExitTime DATETIME,
    IN p_TotalCost DECIMAL(10, 2)
)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        RESIGNAL;
    END;
    START TRANSACTION;
    UPDATE ParkingSessions
    SET ExitTime = p_ExitTime,
        TotalCost = p_TotalCost
    WHERE SessionID = p_SessionID;
    UPDATE ParkingSpots
    SET Status = 'Available'
    WHERE SpotID = (SELECT SpotID FROM ParkingSessions WHERE SessionID =
p_SessionID);
    COMMIT;

```

-- Views --

```

CREATE VIEW TopCustomersByPayment AS
SELECT
    c.CustomerID,
    c.Name,
    SUM(p.Amount) AS TotalAmountPaid
FROM Customers c
JOIN Vehicles v ON c.CustomerID = v.CustomerID
JOIN ParkingSessions ps ON v.VehicleID = ps.VehicleID
JOIN Payments p ON ps.SessionID = p.SessionID
GROUP BY c.CustomerID, c.Name
ORDER BY TotalAmountPaid DESC
LIMIT 10;

```

```

CREATE VIEW VehicleModelCounts AS
SELECT
    Model,
    COUNT(*) AS NumberOfVehicles
FROM Vehicles
GROUP BY Model

```

```
ORDER BY NumberOfVehicles DESC;
```

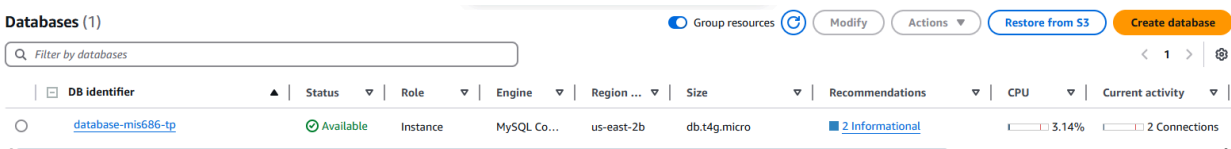
```
CREATE VIEW MostUsedParkingSpots AS
SELECT
    SpotID,
    COUNT(*) AS UsageCount
FROM ParkingSessions
GROUP BY SpotID
HAVING COUNT(*) > 1
ORDER BY UsageCount DESC;
```

```
CREATE VIEW CustomersWithInstantExits AS
SELECT
    c.CustomerID,
    c.Name,
    CASE
        WHEN TIMESTAMPDIFF(HOUR, ps.EntryTime, ps.ExitTime) = 0 THEN '0-1 hour'
        WHEN TIMESTAMPDIFF(HOUR, ps.EntryTime, ps.ExitTime) = 1 THEN '1-2 hours'
        WHEN TIMESTAMPDIFF(HOUR, ps.EntryTime, ps.ExitTime) = 2 THEN '2-3 hours'
        WHEN TIMESTAMPDIFF(HOUR, ps.EntryTime, ps.ExitTime) = 3 THEN '3-4 hours'
    END AS DurationCategory,
    COUNT(*) AS SessionCount
FROM Customers c
JOIN Vehicles v ON c.CustomerID = v.CustomerID
JOIN ParkingSessions ps ON v.VehicleID = ps.VehicleID
WHERE TIMESTAMPDIFF(HOUR, ps.EntryTime, ps.ExitTime) BETWEEN 0 AND 4
GROUP BY c.CustomerID, c.Name, DurationCategory
ORDER BY c.CustomerID, DurationCategory;
```

## Part 4: Database Deployment

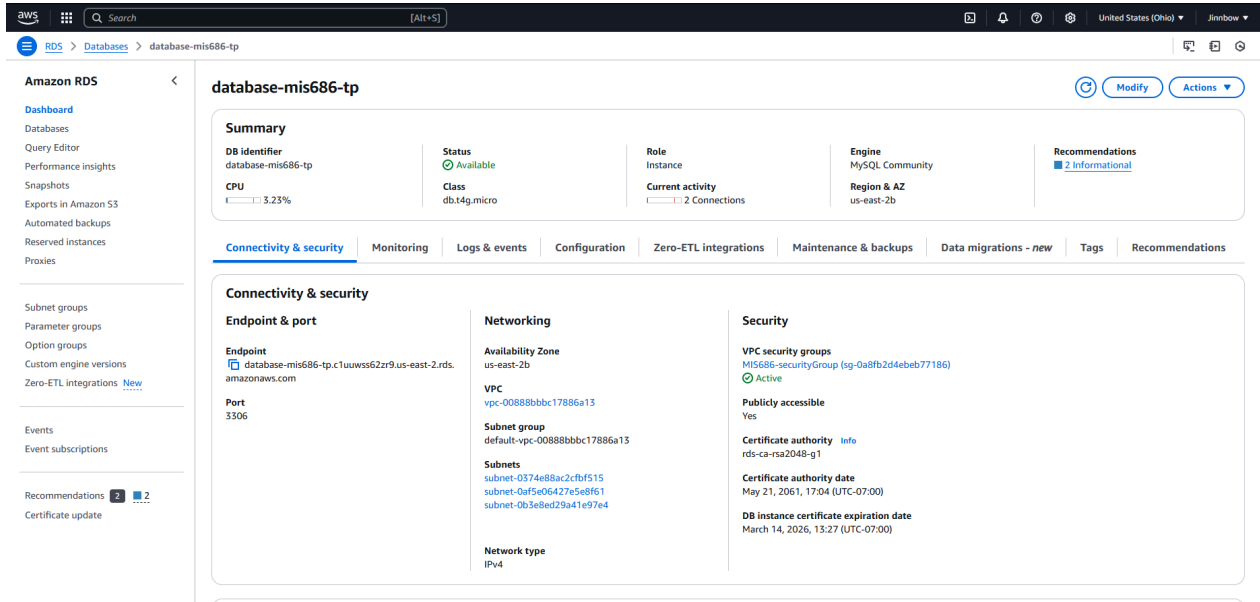
### Database Deployment:

The screenshots below show my deployment of the database on AWS' RDS platform, following the video instructions shown on Canvas, as well as its related configurations. The database's name is "database-mis686-tp".



The screenshot shows the AWS RDS console with a table of database instances. The table has columns for DB identifier, Status, Role, Engine, Region, Size, Recommendations, CPU, and Current activity. One instance is listed: 'database-mis686-tp' with a status of 'Available', role of 'Instance', engine of 'MySQL Co...', region of 'us-east-2b', size of 'db.t4g.micro', and 2 connections. The CPU usage is 3.14%.

DB identifier	Status	Role	Engine	Region	Size	Recommendations	CPU	Current activity
database-mis686-tp	Available	Instance	MySQL Co...	us-east-2b	db.t4g.micro	2 Informational	3.14%	2 Connections



**Amazon RDS**

**database-mis686-tp**

**Summary**

<b>DB identifier</b> database-mis686-tp	<b>Status</b> Available	<b>Role</b> Instance	<b>Engine</b> MySQL Community	<b>Recommendations</b> 2 <a href="#">Informational</a>
<b>CPU</b> 3.23%	<b>Class</b> db.t4g.micro	<b>Current activity</b> 2 Connections	<b>Region &amp; AZ</b> us-east-2b	

**Connectivity & security** | Monitoring | Logs & events | Configuration | Zero-ETL integrations | Maintenance & backups | Data migrations - new | Tags | Recommendations

**Connectivity & security**

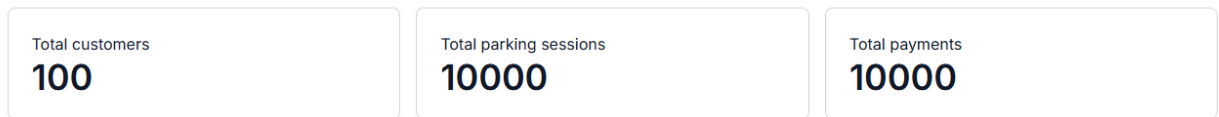
<b>Endpoint &amp; port</b>  <b>Endpoint</b> database-mis686-tp.c1uuwss62zr9.us-east-2.rds.amazonaws.com  <b>Port</b> 3306	<b>Networking</b>  <b>Availability Zone</b> us-east-2b  <b>VPC</b> vpc-00888bbbc17886a13  <b>Subnet group</b> default-vpc-00888bbbc17886a13  <b>Subnets</b> subnet-0374e88ac2c2fbf515 subnet-0af5e06427e5e8f61 subnet-0b3e8ed29a41e97e4  <b>Network type</b> IPv4	<b>Security</b>  <b>VPC security groups</b> MIS686-securityGroup (sg-0a8fb2d4ebeb77186) Active  <b>Publicly accessible</b> Yes  <b>Certificate authority</b> <a href="#">Info</a> rds-ca-rsa2048-g1  <b>Certificate authority date</b> May 21, 2061, 17:04 (UTC-07:00)  <b>DB instance certificate expiration date</b> March 14, 2026, 13:27 (UTC-07:00)
---	--	--

## Part 5: Analytical Questions and Dashboard

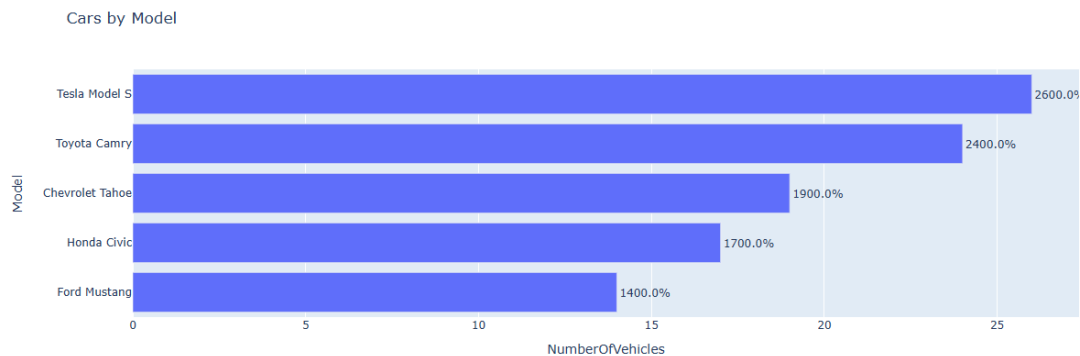
### Analytical Questions:

1. What sort of information can be found from tracking the customers that use the parking garage?
2. Which parking spots are the most frequently used, and how frequently are they reserved?
3. What is the average duration of parking sessions, and does the model of the vehicle have any influence on the parking sessions?
4. What is the total revenue generated by the parking garage over a specific period of time?
5. Which vehicle models are most commonly parked in the garage, and where do they frequently park?
6. Which days of the week have the most parking spots reserved or occupied? Which days have the least?
7. What percentage of the parking spots are occupied, under maintenance, or reserved compared to the number of vacant spots?
8. Which customers have made their reservations, but have not yet accessed the parking garage?
9. Which method of payment is most frequently used by customers?
10. How frequently do customers take shorter parking sessions, and how long do these short sessions tend to last?
11. Which customers take shorter sessions, and how many of these shorter sessions do they take?

## Dashboard Screenshots:



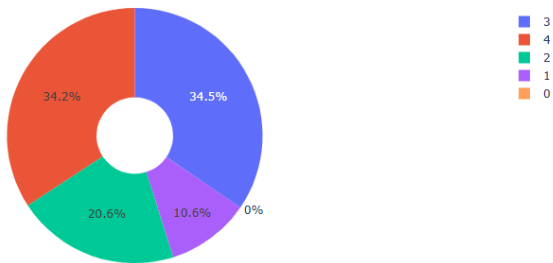
This screenshot shows the top of the dashboard, where it gives three different pieces of information regarding the parking garage. The first is the total number of unique customers, the second is the total number of parking sessions, and the third is the total number of payments. The number of parking sessions and the number of payments should be the same, however the number of customers does not need to match the number of sessions. This data can provide insight to repeat customers over their hundreds of individual parking sessions.



3 rows 2 columns 6 cells			Run SQL Query	Export
	PaymentMethod	PaymentCount		
1	Cash	75		
2	Credit Card	64		
3	Mobile Payment	61		

The screenshot above shows a visualization of questions number 5 and 9. Given the bar graph above labelled "Cars by Model", we can see that of the 100 different customers, 26 of them used the Tesla Model S, while only 14 of them used the Ford Mustang. The data table below it also gives information on how each of them pay, with cash being used to pay 75 times, in contrast to mobile payment only being used 61 times.

Short Sessions & How Long They Last



190 rows3 columns570 cells				Run SQL Query	Export
	CustomerID	Name	DurationHours		
32	16	Customer851	4		
33	18	Customer722	0		
34	18	Customer722	1		
35	18	Customer722	3		
36	19	Customer461	1		
37	19	Customer461	3		
38	20	Customer760	0		
39	20	Customer760	1		
40	23	Customer211	2		
41	23	Customer211	0		
42	23	Customer211	4		

This image shows the next two visualizations, of which answer questions 10 and 11. The pie chart in particular shows an array of sessions that range anywhere from 0 to 4 hours in length, and compares their frequency to each other. Sessions that last 3 or 4 hours are generally quite frequent compared to sessions that last 2 or less hours. The data table below it proves that there are several customers with similar IDs that take multiple sessions for short periods of time, some of which are 0 hours long.

5.7k rows7 columns39.9k cells							Run SQL Query	Export
	CustomerName	CarModel	LicensePlate	ParkingSpot	EntryDate	DurationHours		
1	Customer297	Ford Mustang	JF31	I6	2025-03-16	1		
2	Customer335	Tesla Model S	TB965	I26	2025-03-16	2		
3	Customer570	Chevrolet Tahoe	SG142	D13	2025-03-16	5		
4	Customer297	Ford Mustang	JF31	C6	2025-03-16	8		
5	Customer267	Tesla Model S	UV635	K13	2025-03-16	4		
6	Customer592	Honda Civic	XG618	F18	2025-03-16	13		
7	Customer494	Honda Civic	BA863	S3	2025-03-16	0		
8	Customer652	Chevrolet Tahoe	UD248	E29	2025-03-16	18		
9	Customer570	Chevrolet Tahoe	SG142	F25	2025-03-16	6		
10	Customer26	Toyota Camry	SR234	V30	2025-03-16	6		

This data table is a culmination of all the customers that used the parking garage within the week. It gives a list of information, including their name, license plate number, the parking spot, and how long they spent at that parking spot. This answers question number 1, giving a detailed account of the various information from the customers taking a parking session at the garage. Based on the data collected, parking sessions can range anywhere from 0 hours to more than 24 hours at a time, and the same customer can even park twice on the same day, and in different locations.