

Document report

This report displays the results of the following scans performed on your document: AI detection,

AI RESULTS

Likely Original 90%

Model used: Academic O.O.5

The most accurate AI Detector for ChatGPT, GPT-4o, Gemini Pro, Claude 3.5, Llama 3.1 and other popular AI writing or paraphrasing tools.

How to read the AI score?

This score is a confidence score. If the score is 90%, it should be read as "We are 90% confident that this is AI generated" NOT that 90% of text pasted is AI generated.

AI DETECTION

Level 4 assessment Specification

Certhe Computing Skills for Workplace

Acca4023

Design And Implementation of a Relational Database System

Lecturer: Sadhana Jha, Shah Room

Student ID: 2512924

Student Name: VIORICA-MIHAELA POSIRCA

Table of Contents

1. Introduction	3
2. Scenario Analysis	3
2.1 System Requirements	3
2.2 Main Entities and Attributes	3
3. Entity Relationship Diagram (ERD)	4
4. Database Design and Normalisation	5
4.1 Table Design	5
4.2 Primary and Foreign Keys	5
4.3 Normalisation Process	5
5. Database Implementation in Microsoft Access	6
5.1 Table Creation	6
5.2 Data Population	8
5.3 Forms	11
6. Query Development and Results	13
Query 1: Parking slots currently occupied	13
Query 2: Vehicles parked at a particular time	13
Query 3: Most used parking site	14
Query 4: Vehicle parked in a specific slot	14
Query 5: Total revenue generated per site	15
Query 6: Customers with unpaid bookings	15
Query 7: Daily parking transactions	16
Query 8: Vehicles with the longest parking duration	16
7. Validation and Testing	17
8. Reflection and Challenges	19
9. Conclusion	20
Reference list	21
1. Introduction	

The Smart Parking Management Database System has been created using Microsoft Access which uses an automated database technology and thus provides the opportunity for efficient allocation of vehicles; monitoring and managing parking requests; generating revenues; and managing operations. The latest parking system uses a structured relational database which increases the efficiency of parking allocation by improving dimensions such as traffic congestion and accessibility; and provides a positive customer experience for those who utilise urban transportation systems [1]. By adding database technologies to smart parking solutions, the customer can have better access, reliability or decision-making in the Management of Parking Facilities within the area of parking operations [2].

2. Scenario Analysis

2.1 System Requirements

Using the Smart Parking Management Database System allows for expanded development and improved efficiency of the parking operations at multiple UK parking locations. The system was designed to accommodate functionality for customer registration, vehicle management, parking slot allocation, booking management, and payment processing through a single centralised relational database. Referencing rules were enforced to ensure data quality, and referential integrity was maintained by enforcing validation rules to be applied when entering customer, vehicle, parking slot, booking, and payment information. The system will also require SQL query capabilities for the retrieval of operational reports, including occupied parking slots by vehicle and parking duration for individual bookings, as well as total revenue. Microsoft Access was selected due to being a relational database, along with having user-friendly capabilities for implementing a medium-sized parking management system [3].

2.2 Main Entities and Attributes

The database comprises six major entities; ParkingSite, Customer, Vehicle, ParkingSlot, Booking and Payment, with the main attributes of SiteID, CustomerID, VehicleID, SlotID, BookingID, and PaymentID respectively as Primary Keys. Other attributes supporting the operational elements contained in the database include; vehicle registration number, slot type, booking start & end times, booking payment amount, and payment method [3].

3. Entity Relationship Diagram (ERD)

Figure 1: Entity Relationship Diagram

The Entity Relationship Diagram (ERD) was used as a way to show how different entities interact with each other as part of the rental car parking management system. The ERD will allow you to plan out your database in a structured way; it will create a better understanding of how each entity interacts with one another and what dependencies exist between them [4]. The ERD provides evidence of the one-to-many relationships that exist between ParkingSite and ParkingSlot, Customer and Vehicle, Vehicle and Booking, ParkingSlot and Booking, and Booking and Payment. The purpose of creating the ERD was to provide a way to create an effective management of data to ensure there is no redundancy and to create a way to implement a relational database. Primary keys have been established to uniquely identify each record, while foreign keys create the relationship between tables. This style of database provides the ability to manage bookings, keep track of parking slots and process payments accurately. In addition to assisting in the efficient management of a parking management system, entity relationship modelling will also assist in providing scalability to the database and improving data retrieval from more complex parking management systems [5].

4. Database Design and Normalisation

4.1 Table Design

Relational databases were created for six different tables, including ParkingSite, Customer, Vehicle, ParkingSlot, Booking, and Payment. Each relational table contained the appropriate fields for the operations intended. To have a unique record, the primary key definitions of each table had to use the AutoNumber data type for this to work. For example, Short Text, Number, Currency, and Date/Time are example data types that have been used to accurately store records.

4.2 Primary and Foreign Keys

Primary keys are what is used to uniquely identify a record within each of the relational tables. Additionally, foreign keys create the relationship to connect or be associated to one another and therefore provide the enforcement of the referential integrity of the record. A customer can be identified by their CustomerID within the Vehicle table as being able to connect back to the Customer table. The connection of a VehicleID and SlotID being allocated to a vehicle through the Booking table is done using foreign keys as well.

4.3 Normalisation Process

All of the tables in the database were normalised to the Third Normal Form (3NF) to eliminate redundancy and improve the management of the data dependency. Repeating groups have been eliminated and duplicate data was removed during the normalisation of each of the tables. Every non-key attribute of that relational table is fully dependent on that primary key so it has been logically organised in such a way to provide efficiency for storing the records [6]. Normalising of the database will also help to improve the operational performance and accuracy of the database when executing queries [7].

5. Database Implementation in Microsoft Access

5.1 Table Creation

Figure 2: ParkingSite table

Figure 3: Customer Table

Figure 4: Vehicle Table

Figure 5: ParkingSlot Table

Figure 6: Booking Table

Figure 7: Payment Table

The purpose of the ParkingSite table was to hold various details such as Locations for parking and total amount of slots (capacity) available. The customer table held individual customer personal information such as Customer name, phone number and email address. The vehicle held a record of vehicle registration details for Registered Customers. The ParkingSlot held information about how the slots were assigned and their type of Slot (capacity or parking location). The booking table recorded the time a vehicle was parked, and the time it was removed from parking to get point A to B. The payment table recorded all payment transactions. The Primary Keys/ Foreign Keys were implemented using Microsoft Access's relational tool. In addition to Referential Integrity being enforced, other methods such as Validation Rules for specific information (i.e., phone number formatting, amount of a payment and vehicle registration Number being unique) are in place to ensure a consistent and accurate data set throughout the related tables.

5.2 Data Population

Figure 8: ParkingSite Table data

Figure 9: Customer Table data

Figure 10: Vehicle Table data

Figure 11: ParkingSlot Table data

Figure 12: Booking Table data

Figure 13: Payment Table data

To realistically simulate the operation of an effective parking management system, each table in the database contained ten records from the UK. Examples of parking location sites include cities such as London, Manchester, Birmingham and Liverpool. Likewise, the records generated for customers and vehicles used UK-style phone numbers and vehicle registration plate formats. Booking records stored details of time booked for parking as well as details of payments. The order of insertion into each table was based on their relational dependency so as to ensure that the insertion of a record into one table did not violate the referential integrity of any other related table. Parent tables such as ParkingSite and Customer were populated first followed by their related child tables - Vehicle and Booking to establish the valid relationships and prevent invalid records from being inserted into any relevant table.

5.3 Forms

Figure 14: Customer Registration Form

Figure 15: Vehicle Form

Figure 16: Booking Form

Figure 17: Payment Form

The purpose of creating forms was to enhance the usability of the database and to streamline interactions between users and the database. The forms provided by Microsoft Access Form tools included Customer Registration, Vehicle, Booking and Payment Forms. Forms used combo boxes so that users could reduce the possibility of making errors when entering data, as well as entering data in a more efficient manner when creating a record. The forms allowed users to navigate the records of the database easily while maintaining relational integrity and consistency through the use of linked data for selection.

6. Query Development and Results

The implementation of queries provides a means to perform the retrieval of information from the database efficiently, produce reports, and analyse parking management [8].

Query 1: Parking slots currently occupied

Query 2: Vehicles parked at a particular time

Query 3: Most used parking site

Query 4: Vehicle parked in a specific slot

Query 1 finds all currently occupied parking spaces based on whether or not they have space available. Query 2 retrieves all vehicles parked within a specified time frame using the booking timestamp. Query 3 counts the total number of bookings, by location, in order to find out which location is the most frequently used for parking. Query 4 retrieves information about all vehicles that have been parked at a specific parking location.

Query 5: Total revenue generated per site

Query 6: Customers with unpaid bookings

Query 7: Daily parking transactions

Query 8: Vehicles with the longest parking duration

Through Queries 5 through 8, aggregate payment calculations generated total parking site revenue (Query 5), listed customer and vehicle details related to each booking record (Query 6), showed parking transactions and payment method/amounts (Query 7), and identified vehicles with the highest total days parked using the DateDiff function (Query 8). Relational joins, filtering operations, aggregate functions, sorting, and calculated fields were all used to complete the developed queries. Query optimisation enhances both the data retrieval performance and the operational reporting efficiency of a relational database [9]. In addition, using SQL effectively contributes to a more efficient analysis of transactions and assists with decision-making for parking management.

7. Validation and Testing

Test 1: Referential Integrity

While adding new vehicle data with Customer ID 99, due to referential integrity constraints the database does not allow adding the data into the Vehicle column and directs users to enter in the customer table first as there is no such customer exists in the table with ID 99.

Test 2: Phone Validation Rule

In this case while adding a new Customer in the Customer table the database prompts an error message due to inconsistent input in the Phone number which is not in the UK format (Starts with 07) and this ensures validation rule being met.

Test 3: Duplicate Vehicle Registration

In this case when trying to insert a new vehicle with customer ID 4 but duplicate Registration number as QW19ERT which is the registration number of Customer with ID 10, the database does not allow the new customer insertion into the database. This shows that the database does not allow duplicate entry in the unique fields.

Test 4: Payment Validation

This shows another validation method as while entering payment details with negative payment amount (-10), the database rejects the entry as the validation rule is mentioned as the payment amount needs to be greater than zero. The validation processes have improved data consistency, integrity, and operational reliability in the parking management database system. Validation mechanisms within databases play a vital role in providing accurate records of transactions; they also help to minimise operational errors resulting from automated systems [10].

8. Reflection and Challenges

Creating a Smart Parking Management Database System allowed me to learn how to use Microsoft Access to create relational databases. Creating relationships between two tables and making sure the primary key number matched the foreign key number in both tables were two of the biggest challenges I had. I ran into mismatch errors between the sizes of the primary and foreign keys on some tables when I tried establishing relationships between two tables. I encountered problems when creating SQL queries because the Microsoft Access date field format did not match the date field format used by the query I was trying to run, resulting in blank records being returned when I executed a query for each of the date fields returned a record. After repeatedly testing the table relationships and my SQL queries, I was able to better understand both query design and referential integrity and develop better validation rules for all of the tables used in the database [11]. The process of creating forms and using combo boxes further improved my ability to design user interfaces in Access that are easy to use. Overall, the entire project gave me greater confidence in my ability to design databases, normalise data, develop SQL queries and manage databases.

9. Conclusion

The Smart Parking Management System demonstrated an effective way of implementing a relational database to efficiently manage parking operations. The system was able to manage customer information, vehicle information, parking slot assignments, reservations, and payment transactions within the confines of an organised database. The implementation included relationships between tables, validation rules, SQL statements (queries), and forms which yielded improvements in data consistency, data usability and operational efficiency. This project will provide a better understanding of database design, normalisation, and query processing with Microsoft's Access program.

Reference list

- [1] A. Fahim, M. Hasan, and M. A. Chowdhury, "Smart parking systems: comprehensive review based on various aspects," *Heliyon*, vol. 7, no. 5, pp. 1–21, May 2021, doi: <https://doi.org/10.1016/j.heliyon.2021.e07050>.
- [2] M. Alymani, L. A. Almoqhem, D. A. Alabdulwahab, A. A. Alghamdi, and K. Raza, "Enabling smart parking for smart cities using Internet of Things (IoT) and machine learning," *PeerJ Computer Science*, vol. 11, no. 21, pp. 1–27, Jan. 2025, doi: <https://doi.org/10.7717/peerj-cs.2544>.
- [3] M. Adil, "THE POSITIVE IMPACT OF MICROSOFT ACCESS IN EDUCATION AND ITS INTEGRATION WITH CLOUD TECHNOLOGIES: AN ACTIVITY-ORIENTED PEDAGOGICAL APPROACH," *European Journal of Education Studies*, vol. 13, no. 1, pp. 1–10, 2026, doi: <https://doi.org/10.46827/ejes.v13i1.6484>.
- [4] S. Al-Fedaghi, "Conceptual Entity-Relationship Model: Underneath the Simplicity and Staticity," *arXiv.org*, pp. 1–17, 2025, Accessed: Mar. 24, 2025. [Online]. Available: <https://arxiv.org/abs/2503.06155>
- [5] P. Skavantzios and S. Link, "Entity/Relationship Modeling for Property Graphs," 2025. Accessed: May 28, 2026. [Online]. Available: https://ceur-ws.org/Vol-4099/ER25_tutorial_Skavantzios.pdf
- [6] K. Kumar and S. Kumar, "Relational Database Normalization under Tabular Approach: A Design Methodology," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, pp. 2160–2164, 2017, doi: <https://doi.org/10.26483/ijarcs.v8i5.4041>.
- [7] W. Ajayi, K. Okesola, D. Nteziyayo, F. Odo, and A. U. Akpan, "Relational Database Normalization vs Denormalization: A Performance Perceptive," *Asian Journal of Electrical Sciences*, vol. 14, no. 2, pp. 53–66, Dec. 2025, doi: <https://doi.org/10.70112/ajes-2025.14.2.4291>.

- [8] A. Sandhya and G. Y. Mythili, "SQL Optimization for Fast Retrieval of Transportation Route Data in TMS," *Boletim da Sociedade Paranaense de Matemática*, vol. 44, pp. 1–9, 2026, doi: <https://doi.org/10.5269/bspm.81609>.
- [9] M. Abbasi, M. V. Bernardo, P. Váz, J. Silva, and P. Martins, "Optimizing Database Performance in Complex Event Processing through Indexing Strategies," *Data*, vol. 9, no. 8, pp. 1–24, Jul. 2024, doi: <https://doi.org/10.3390/data9080093>.
- [10] A. K. Thomer and K. M. Wickett, "Relational data paradigms: What do we learn by taking the materiality of databases seriously?," *Big Data & Society*, vol. 7, no. 1, pp. 1–16, Jan. 2020, doi: <https://doi.org/10.1177/2053951720934838>.
- [11] Q. Abbas, G. Ahmad, T. Alyas, T. Alghamdi, Y. Alsaawy, and A. Alzahrani, "Revolutionizing Urban Mobility: IoT-Enhanced Autonomous Parking Solutions with Transfer Learning for Smart Cities," *Sensors*, vol. 23, no. 21, pp. 1–15, Jan. 2023, doi: <https://doi.org/10.3390/s23218753>.

Appendices

Appendix 1: SQL Queries

Q1

```
SELECT ParkingSlot.SlotNumber, ParkingSlot.SlotType, ParkingSlot.AvailabilityStatus
FROM ParkingSlot
WHERE ParkingSlot.AvailabilityStatus="Occupied";
```

Q2

```
SELECT Vehicle.RegistrationNumber, Booking.EntryTime, Booking.ExitTime
FROM Vehicle INNER JOIN Booking ON Vehicle.VehicleID=Booking.VehicleID
WHERE Booking.EntryTime Between #6/1/2026 8:0:0# And #6/5/2026 18:0:0#
ORDER BY Booking.EntryTime;
```

Q3

```
SELECT ParkingSite.SiteName, Count(Booking.BookingID) AS TotalBookings
FROM (ParkingSite INNER JOIN ParkingSlot ON ParkingSite.SiteID = ParkingSlot.SiteID) INNER JOIN Booking ON ParkingSlot.SlotID =
Booking.SlotID
GROUP BY ParkingSite.SiteName
ORDER BY Count(Booking.BookingID) DESC;
```

Q4

```
SELECT ParkingSlot.SlotNumber, Vehicle.RegistrationNumber, Booking.EntryTime
FROM (ParkingSlot INNER JOIN Booking ON ParkingSlot.SlotID = Booking.SlotID) INNER JOIN Vehicle ON Booking.VehicleID =
Vehicle.VehicleID
WHERE ParkingSlot.SlotNumber="A01";
```

Q5

```
SELECT ParkingSite.SiteName, Sum(Payment.PaymentAmount) AS TotalRevenue
FROM ((ParkingSite INNER JOIN ParkingSlot ON ParkingSite.SiteID=ParkingSlot.SiteID) INNER JOIN Booking ON
ParkingSlot.SlotID=Booking.SlotID) INNER JOIN Payment ON Booking.BookingID=Payment.BookingID
GROUP BY ParkingSite.SiteName
ORDER BY Sum(Payment.PaymentAmount) DESC;
```

Q6

```
SELECT Customer.FirstName, Customer.LastName, Vehicle.RegistrationNumber, Vehicle.VehicleType
FROM Customer INNER JOIN Vehicle ON Customer.CustomerID = Vehicle.CustomerID;
```

Q7

```
SELECT Vehicle.RegistrationNumber, Payment.PaymentAmount, Payment.PaymentMethod, Payment.PaymentDate
FROM (Vehicle INNER JOIN Booking ON Vehicle.VehicleID=Booking.VehicleID) INNER JOIN Payment ON
Booking.BookingID=Payment.BookingID;
```

Q8

```
SELECT Vehicle.RegistrationNumber, Booking.EntryTime, Booking.ExitTime, DateDiff("h",[EntryTime],[ExitTime]) AS ParkingHours
FROM Vehicle INNER JOIN Booking ON Vehicle.VehicleID = Booking.VehicleID
ORDER BY DateDiff("h",[EntryTime],[ExitTime]) DESC;
```