CCCS 215 Database Management

Section: W1

Final Report

Made by:

- Razan alghanmi 2111007
- Shahad Kulaibi 2114565
- Elaf Salem 2110527

Instructor: Dr.Azhar Melabari

Project schedule

Task	Due date	Responsible member
Project proposal	22 Sep	All members
DB Analysis	8 Oct	All members
DB Design(ER model)	8 Oct	All members
DB Design(Normalization and Mapping)	30 Oct	All members
DB Implementation and testing	5 December	All members

Mall Navigation Application Database

Problem:

The problem is that navigating through a large shopping mall can be confusing and time-consuming for visitors. It is often challenging to find specific stores, locate amenities such as restrooms or food courts, and efficiently navigate from one area to another. This can result in frustration and wasted time for shoppers.

Solution:

The solution is to create a database-driven mall navigation system that provides visitors with an easy and efficient way to navigate through the shopping mall. The system will utilize a combination of digital maps, location tracking, and real-time data to guide users to their desired destinations within the mall.

List of entities:

Mall
Customer
Car
Service
Store

Entity	Description	Primary Key
Mall	This table can be used to store and retrieve details about various malls, including their locations, contact information, and operating hours.	Mall_ID
Customer_Visit	establishes a many-to-many relationship between the "Customer" and "Mall" tables. allowing for the association of specific customers with specific malls and the date of their visit.	Customer_ID Mall_ID
Customer	establishes a many-to-many relationship between the "Customer" and "Mall" tables. allowing for the association of specific customers with specific malls and the date of their visit.	Customer_ID
Car	contains information about customers' cars.	Car_ID
Parking_Car	contains information about parking usage by customers' cars at the malls. This table can be used to store and retrieve details about customers' parking usage at specific malls.	Parking_Number Date_Of_Use Time_Of_Use
Store	contains information about the stores located within the malls. This table can be used to store and retrieve details about the various stores within the malls.	Store_ID
Mall_Store	establishes a many-to-many relationship between the "Mall" and "Store" tables. allowing for the association of specific stores with specific malls.	Store_ID Mall_ID
Service	contains information about the stores located within the malls. This table can be used to store and retrieve details	Service_ID

	about the various stores within the malls.	
Mall_Service	establishes a many-to-many relationship between the "Mall" and "Service" tables. allowing for the association of specific services with specific malls.	Mall_ID Service_ID

The database structure supports the management of mall details, customer interactions, parking usage, store information, and available services within the malls. It enables tracking of customer visits, association of stores and services with specific malls, and capturing parking details for customers' cars. This structured approach facilitates the development of a comprehensive mall navigation app database to enhance user experience and engagement within the mall environment.

Business Rules:

1. Mall - Mall_Service: One-to-Many (One mall can have multiple services, but each service is associated with only one mall)

2. Service - Mall_Service: One-to-Many (One service can be offered in multiple malls, but each mall can offer multiple services)

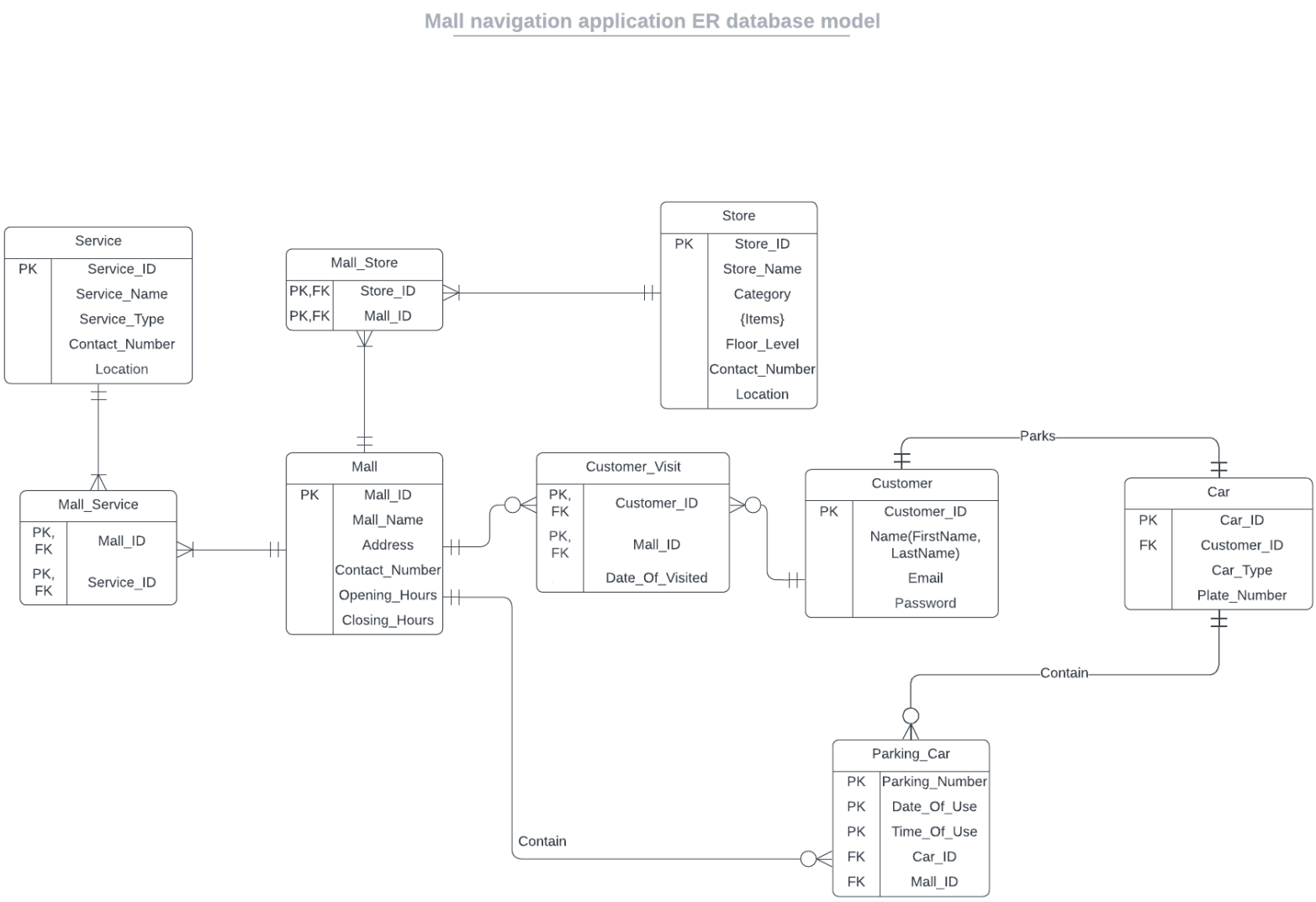
3. Store - Store_Item: One-to-Many (One store can have multiple items, but each item is associated with only one store)

4. Store - Mall_Store: Many-to-Many (One store can be located in multiple malls, and one mall can have multiple stores)

5. Customer - Customer_Visit: One-to-Many (One customer can visit multiple malls, but each visit is associated with only one customer)

6. Car - Parking_Car: One-to-Many (One car can be parked multiple times, but each parking record is associated with only one car)

ER – Model:


Mall navigation application ER database model
Service
PK
Service_ID
Service_Name
Service_Type
Contact_Number
Location
Mall_Store
PK,FK
Store_ID
PK,FK
Mall_ID
Store
PK
Store_ID
Store_Name
Category
{Items}
Floor_Level
Contact_Number
Location
Mall_Service
PK, FK
Mall_ID
PK, FK
Service_ID
Mall
PK
Mall_ID
Mall_Name
Address
Contact_Number
Opening_Hours
Closing_Hours
Customer_Visit
PK, FK
Customer_ID
PK, FK
Mall_ID
Date_Of_Visited
Parks
Customer
PK
Customer_ID
Name(FirstName, LastName)
Email
Password
Car
PK
Car_ID
FK
Customer_ID
Car_Type
Plate_Number
Contain
Parking_Car
PK
Parking_Number
PK
Date_Of_Use
PK
Time_Of_Use
FK
Car_ID
FK
Mall_ID
Contain

Normalization:

To normalize the schema to the third normal form (3NF), we need to eliminate any transitive dependencies and ensure that each attribute directly depends on the primary key.

1NF:

The "Building" relation is not in the first normal form (1NF) as it exhibits a unique primary key (PK) that is non-null, but not all its attributes are atomic. There are multi-valued attributes or repeated groups in the schema.

1. Store ((pk) store_id, store_name, category, {items}, floor_level, contact_number, location)

- Remove the multi-valued attribute {items} and create a separate table StoreItems to eliminate the transitive dependency.

  - StoreItems ((fk) storeid, item)

- Now the Store table will become:

  - Store ((pk) storeid, storename, category, floorlevel, contactnumber, location)

2. Customer ((pk) customer_id, name (FirstName, LastName), Email, Password)

- Remove the multivalued attribute and create two separate attributes.

- Now the Customer table will become:

  - Customer ((pk) customer_id, FirstName, LastName, Email, Password)

2NF:

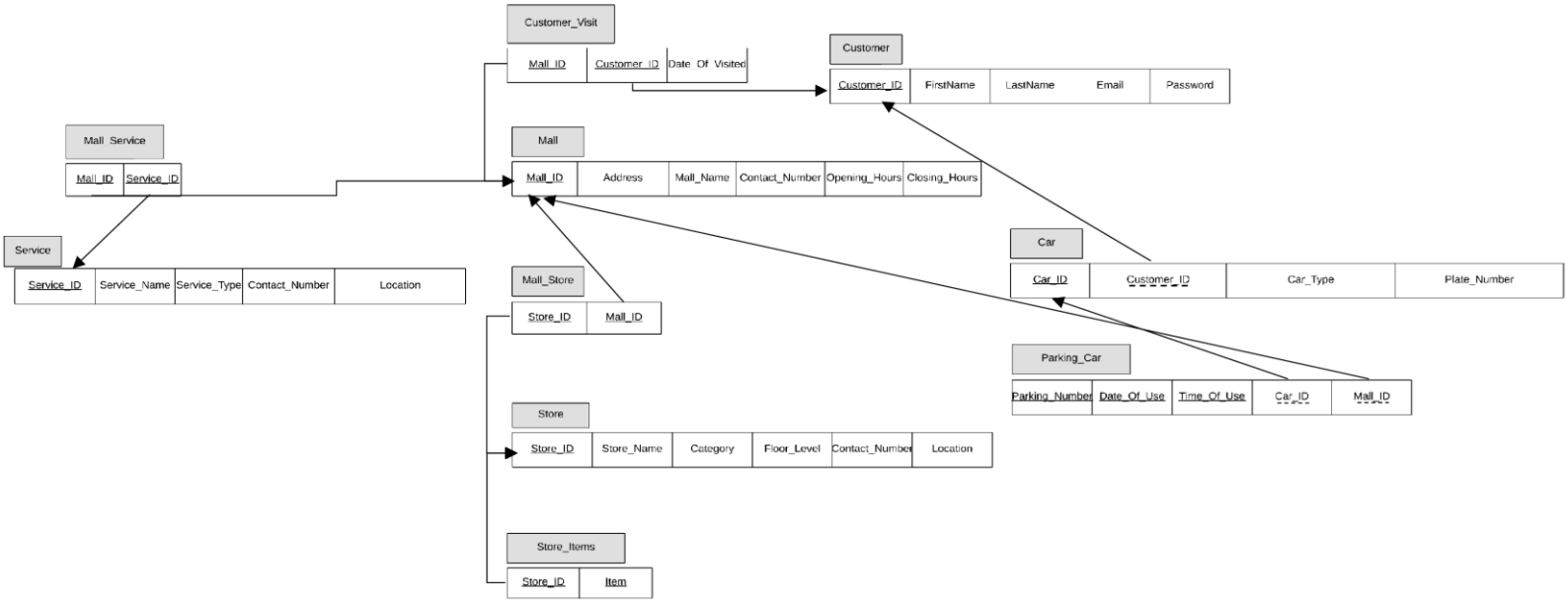
The "Building" relation already adheres to the second normal form (2NF) prerequisites since it lacks partial dependencies. Every attribute is fully functionally dependent on all components of the primary key, meeting the criteria for 2NF.

3NF:

To fulfill the third normal form (3NF), the focus is on eliminating transitive dependencies. All non-key attributes must be solely and fully functionally dependent on the primary key. The schema is confirmed to meet the rules of 3NF, ensuring there are no non-key attributes that transitively depend on the primary key.

All the tables have been normalized to the third normal form (3NF) by removing any transitive dependencies and ensuring that each attribute is directly dependent on the primary key.

Mapping:



Functional Dependencies
Mall_ID → Address, Mall_Name, Contact_Number, Opening_Hours, Closing_Hours
Customer_ID → FirstName, LastName, Email, Password, Date_Of_Visit
Store_ID → Store_Name, Category, Floor_Level, Contact_Number, Location, Item
Service_ID → Service_Name, Service_Type, Contact_Number, Location
Car_ID → Car_Type, Plate_Number

Table creation and row insertion:

```sql
CREATE TABLE Mall (
    Mall_ID INT PRIMARY KEY,
    Mall_Name VARCHAR2(50),
    Address VARCHAR2(100),
    Contact_Number VARCHAR2(20),
    Opening_Hours VARCHAR2(20),
    Closing_Hours VARCHAR2(20)
);

INSERT INTO Mall (Mall_ID, Mall_Name, Address, Contact_Number, Opening_Hours, Closing_Hours)
VALUES (1, 'Mall A', '123 Main Street', '123-456-7890', '9:00 AM', '9:00 PM');

INSERT INTO Mall (Mall_ID, Mall_Name, Address, Contact_Number, Opening_Hours, Closing_Hours)
VALUES (2, 'Mall B', '456 Elm Street', '987-654-3210', '8:00 AM', '10:00 PM');

INSERT INTO Mall (Mall_ID, Mall_Name, Address, Contact_Number, Opening_Hours, Closing_Hours)
VALUES (3, 'Mall C', '789 Oak Street', '555-123-4567', '10:00 AM', '8:00 PM');

INSERT INTO Mall (Mall_ID, Mall_Name, Address, Contact_Number, Opening_Hours, Closing_Hours)
VALUES (4, 'Mall D', '321 Maple Street', '222-333-4444', '9:30 AM', '9:30 PM');

INSERT INTO Mall (Mall_ID, Mall_Name, Address, Contact_Number, Opening_Hours, Closing_Hours)
VALUES (5, 'Mall E', '987 Pine Street', '111-222-3333', '10:00 AM', '10:00 PM');


CREATE TABLE Service (
    Service_ID INT PRIMARY KEY,
    Service_Name VARCHAR2(50),
    Service_Type VARCHAR2(50),
    Contact_Number VARCHAR2(20),
    Location VARCHAR2(150)
);

INSERT INTO Service (Service_ID, Service_Name, Service_Type, Contact_Number, Location)
VALUES (1, 'Yota', 'Restaurant', '555-123-4567', 'Yota Restaurant is located on the ground floor, near the main entrance.');

INSERT INTO Service (Service_ID, Service_Name, Service_Type, Contact_Number, Location)
VALUES (2, 'Bito', 'Restaurant', '987-654-3210', 'Bito Restaurant is situated on the first floor, overlooking the central atrium.');

INSERT INTO Service (Service_ID, Service_Name, Service_Type, Contact_Number, Location)
VALUES (3, 'Elevator001', 'Elevator', '123-456-7890', 'Elevator001 is located near the main lobby, providing access to all floors of the building.')

INSERT INTO Service (Service_ID, Service_Name, Service_Type, Contact_Number, Location)
VALUES (4, 'Elevator002', 'Elevator', '222-333-4444', 'Elevator002 is positioned at the rear of the building, serving the upper levels.');

INSERT INTO Service (Service_ID, Service_Name, Service_Type, Contact_Number, Location)
VALUES (5, 'W.C', 'W.C', '111-222-3333', 'The W.C facilities are conveniently located on the ground floor, adjacent to the food court.' );

INSERT INTO Service (Service_ID, Service_Name, Service_Type, Contact_Number, Location)
VALUES (6, 'FitnessFusion', 'Gym', '333-222-1111', 'FitnessFusion can be found on the third floor, equipped with state-of-the-art exercise machines

INSERT INTO Service (Service_ID, Service_Name, Service_Type, Contact_Number, Location)
VALUES (7, 'KidsZone', 'Play Area', '999-888-7777', 'KidsZone is situated near the food court, providing a safe and enjoyable play area for children

INSERT INTO Service (Service_ID, Service_Name, Service_Type, Contact_Number, Location)
VALUES (8, 'SecurityHub', 'Security Office', '000-111-2222', 'SecurityHub is centrally located, overseeing the safety and security of the entire mal



CREATE TABLE Mall_Service (
    Mall_ID INT,
    Service_ID INT,
    PRIMARY KEY (Mall_ID, Service_ID),
    FOREIGN KEY (Mall_ID) REFERENCES Mall(Mall_ID),
    FOREIGN KEY (Service_ID) REFERENCES Service(Service_ID)
);

INSERT INTO Mall_Service (Mall_ID, Service_ID)
VALUES (1, 1);
INSERT INTO Mall_Service (Mall_ID, Service_ID)
VALUES (1, 2);
INSERT INTO Mall_Service (Mall_ID, Service_ID)
VALUES (2, 3);
INSERT INTO Mall_Service (Mall_ID, Service_ID)
VALUES (3, 4);
INSERT INTO Mall_Service (Mall_ID, Service_ID)
```

```
VALUES (3, 5);
INSERT INTO Mall_Service (Mall_ID, Service_ID)
VALUES (4, 1);
INSERT INTO Mall_Service (Mall_ID, Service_ID)
VALUES (4, 6);
INSERT INTO Mall_Service (Mall_ID, Service_ID)
VALUES (5, 1);
INSERT INTO Mall_Service (Mall_ID, Service_ID)
VALUES (5, 7);
INSERT INTO Mall_Service (Mall_ID, Service_ID)
VALUES (5, 8);

CREATE TABLE Store (
    Store_ID INT PRIMARY KEY,
    Store_Name VARCHAR2(50),
    Category VARCHAR2(50),
    Floor_Level INT,
    Contact_Number VARCHAR2(20),
    Location VARCHAR2(150)
);

INSERT INTO Store (Store_ID, Store_Name, Category, Floor_Level, Contact_Number, Location)
VALUES (1, 'Runway fashion', 'Clothing', 1, '555-123-4567', 'Runway fashion is located on the second floor, near the food court.');

INSERT INTO Store (Store_ID, Store_Name, Category, Floor_Level, Contact_Number, Location)
VALUES (2, 'Tech Oasis', 'Electronics', 2, '987-654-3210', 'Tech Oasis is located on the second floor, near the electronic accessories section.');

INSERT INTO Store (Store_ID, Store_Name, Category, Floor_Level, Contact_Number, Location)
VALUES (3, 'Bookworm', 'Books', 1, '123-456-7890','Bookworm is located on the ground floor, next to the kids play area');

INSERT INTO Store (Store_ID, Store_Name, Category, Floor_Level, Contact_Number, Location)
VALUES (4, 'Glamour Boutique', 'Jewelry', 3, '222-333-4444','Glamour Boutique is located on the first floor, near the escalators.');

INSERT INTO Store (Store_ID, Store_Name, Category, Floor_Level, Contact_Number, Location)
VALUES (5, 'Shoes zone', 'Shoes', 2, '111-222-3333', 'Shoes zone is located on the third floor, near the cinema.');

INSERT INTO Store (Store_ID, Store_Name, Category, Floor_Level, Contact_Number, Location)
VALUES (6, 'PetParadise', 'Pet Store', 3, '222-333-4444', 'PetParadise is situated on the ground floor, offering pet supplies and grooming services

INSERT INTO Store (Store_ID, Store_Name, Category, Floor_Level, Contact_Number, Location)
VALUES (7, 'TechRepair', 'Electronics Repair', 3, '777-999-1111', 'TechRepair is situated on the ground floor, providing repair services for electro

INSERT INTO Store (Store_ID, Store_Name, Category, Floor_Level, Contact_Number, Location)
VALUES (8, 'BeautyBoutique', 'Cosmetics Store', 1, '333-444-5555', 'BeautyBoutique is positioned on the first floor, offering a variety of beauty ar

CREATE TABLE Store_Item (
    Store_ID INT,
    Item VARCHAR2(50),
    PRIMARY KEY (Store_ID, Item),
    FOREIGN KEY (Store_ID) REFERENCES Store(Store_ID)
);

INSERT INTO Store_Item (Store_ID, Item)
VALUES (1, 'Dress');
INSERT INTO Store_Item (Store_ID, Item)
VALUES (1, 'Jeans');

INSERT INTO Store_Item (Store_ID, Item)
VALUES (2, 'Laptop');

INSERT INTO Store_Item (Store_ID, Item)
VALUES (3, 'Introduction to database managment');

INSERT INTO Store_Item (Store_ID, Item)
VALUES (4, 'Earrings');
INSERT INTO Store_Item (Store_ID, Item)
VALUES (4, 'Necklace');

INSERT INTO Store_Item (Store_ID, Item)
VALUES (5, 'Sandles');
```

```sql
INSERT INTO Store_Item (Store_ID, Item)
VALUES (6, 'Dry food');
INSERT INTO Store_Item (Store_ID, Item)
VALUES (6, 'Toy ball');

INSERT INTO Store_Item (Store_ID, Item)
VALUES (7, 'Protection screen');

INSERT INTO Store_Item (Store_ID, Item)
VALUES (8 , 'Lipstick');
INSERT INTO Store_Item (Store_ID, Item)
VALUES (8 , 'Eyeliner');
INSERT INTO Store_Item (Store_ID, Item)
VALUES (8 , 'Mascara');

CREATE TABLE Mall_Store (
    Store_ID INT,
    Mall_ID INT,
    PRIMARY KEY (Store_ID, Mall_ID),
    FOREIGN KEY (Store_ID) REFERENCES Store(Store_ID),
    FOREIGN KEY (Mall_ID) REFERENCES Mall(Mall_ID)
);

INSERT INTO Mall_Store (Mall_ID, Store_ID)
VALUES (1, 1);
INSERT INTO Mall_Store (Mall_ID, Store_ID)
VALUES (1, 2);
INSERT INTO Mall_Store (Mall_ID, Store_ID)
VALUES (2, 3);
INSERT INTO Mall_Store (Mall_ID, Store_ID)
VALUES (3, 4);
INSERT INTO Mall_Store (Mall_ID, Store_ID)
VALUES (3, 5);
INSERT INTO Mall_Store (Mall_ID, Store_ID)
VALUES (4, 1);
INSERT INTO Mall_Store (Mall_ID, Store_ID)
VALUES (4, 6);
INSERT INTO Mall_Store (Mall_ID, Store_ID)
VALUES (5, 1);
INSERT INTO Mall_Store (Mall_ID, Store_ID)
VALUES (5, 7);
INSERT INTO Mall_Store (Mall_ID, Store_ID)
VALUES (5, 8);

CREATE TABLE Customer (
    Customer_ID INT PRIMARY KEY,
    FirstName VARCHAR2(50),
    LastName VARCHAR2(50),
    Email VARCHAR2(100),
    Password VARCHAR2(100)
);
```

```sql
INSERT INTO Customer (Customer_ID, FirstName, LastName, Email, Password)
VALUES (1, 'John', 'Doe', 'john.doe@example.com', 'password123');

INSERT INTO Customer (Customer_ID, FirstName, LastName, Email, Password)
VALUES (2, 'Jane', 'Smith', 'jane.smith@example.com', 'p@ssw0rd');

INSERT INTO Customer (Customer_ID, FirstName, LastName, Email, Password)
VALUES (3, 'David', 'Johnson', 'david.johnson@example.com', 'secret123');

INSERT INTO Customer (Customer_ID, FirstName, LastName, Email, Password)
VALUES (4, 'Emily', 'Wilson', 'emily.wilson@example.com', 'ilovecats');

INSERT INTO Customer (Customer_ID, FirstName, LastName, Email, Password)
VALUES (5, 'Michael', 'Brown', 'michael.brown@example.com', 'qwerty123');

CREATE TABLE Customer_Visit (
    Customer_ID INT,
    Mall_ID INT,
    Date_Of_Visited DATE            ,
    FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID),
    FOREIGN KEY (Mall_ID) REFERENCES Mall(Mall_ID)
);

INSERT INTO Customer_Visit (Customer_ID, Mall_ID, Date_Of_Visited)
VALUES (1, 1, TO_DATE('2023-12-04', 'YYYY-MM-DD'));
```

```sql
INSERT INTO Customer_Visit (Customer_ID, Mall_ID, Date_Of_Visited)
VALUES (2, 2, TO_DATE('2023-12-04', 'YYYY-MM-DD'));

INSERT INTO Customer_Visit (Customer_ID, Mall_ID, Date_Of_Visited)
VALUES (3, 3, TO_DATE('2023-12-04', 'YYYY-MM-DD'));

INSERT INTO Customer_Visit (Customer_ID, Mall_ID, Date_Of_Visited)
VALUES (4, 4, TO_DATE('2023-12-04', 'YYYY-MM-DD'));

INSERT INTO Customer_Visit (Customer_ID, Mall_ID, Date_Of_Visited)
VALUES (5, 5, TO_DATE('2023-12-04', 'YYYY-MM-DD'));

CREATE TABLE Car (
    Car_ID INT PRIMARY KEY,
    Customer_ID INT,
    Car_Type VARCHAR2(50),
    Plate_Number VARCHAR2(50),
    FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)
);

INSERT INTO Car (Car_ID, Customer_ID, Car_Type, Plate_Number)
VALUES (1, 1, 'Mercedes-Benz C-Class', 'ABC-123');

INSERT INTO Car (Car_ID, Customer_ID, Car_Type, Plate_Number)
VALUES (2, 2, 'Toyota Camry', 'XYZ-789');

INSERT INTO Car (Car_ID, Customer_ID, Car_Type, Plate_Number)
VALUES (3, 3, 'Ford Mustang', 'LMN-456');

INSERT INTO Car (Car_ID, Customer_ID, Car_Type, Plate_Number)
VALUES (4, 4, 'Jeep Wrangler', 'PQR-321');

INSERT INTO Car (Car_ID, Customer_ID, Car_Type, Plate_Number)
VALUES (5, 5, 'BMW X5', 'JKL-987');
CREATE TABLE Parking_Car (
    Parking_Number INT,
    Date_Of_Use DATE,
    Time_Of_Use TIMESTAMP,
    Car_ID INT,
    Mall_ID INT,
    PRIMARY KEY (Parking_Number, Date_Of_Use, Time_Of_Use),
    FOREIGN KEY (Car_ID) REFERENCES Car(Car_ID),
    FOREIGN KEY (Mall_ID) REFERENCES Mall(Mall_ID)
);

INSERT INTO Parking_Car (Parking_Number, Date_Of_Use, Time_Of_Use, Car_ID, Mall_ID)
VALUES (1, TO_DATE('2023-12-04', 'YYYY-MM-DD'), TO_TIMESTAMP('09:00:00', 'HH24:MI:SS'), 1, 1);

INSERT INTO Parking_Car (Parking_Number, Date_Of_Use, Time_Of_Use, Car_ID, Mall_ID)
VALUES (1, TO_DATE('2023-12-04', 'YYYY-MM-DD'), TO_TIMESTAMP('11:00:00', 'HH24:MI:SS'), 1, 1);

INSERT INTO Parking_Car (Parking_Number, Date_Of_Use, Time_Of_Use, Car_ID, Mall_ID)
VALUES (2, TO_DATE('2023-12-04', 'YYYY-MM-DD'), TO_TIMESTAMP('10:30:00', 'HH24:MI:SS'), 2, 2);
INSERT INTO Parking_Car (Parking_Number, Date_Of_Use, Time_Of_Use, Car_ID, Mall_ID)
VALUES (3, TO_DATE('2023-12-04', 'YYYY-MM-DD'), TO_TIMESTAMP('11:00:00', 'HH24:MI:SS'), 3, 3);

INSERT INTO Parking_Car (Parking_Number, Date_Of_Use, Time_Of_Use, Car_ID, Mall_ID)
VALUES (4, TO_DATE('2023-12-04', 'YYYY-MM-DD'), TO_TIMESTAMP('12:30:00', 'HH24:MI:SS'), 4, 4);

INSERT INTO Parking_Car (Parking_Number, Date_Of_Use, Time_Of_Use, Car_ID, Mall_ID)
VALUES (5, TO_DATE('2023-12-04', 'YYYY-MM-DD'), TO_TIMESTAMP('10:05:00', 'HH24:MI:SS'), 5, 5);
```

Quires:

1 - This query retrieves the mall name, service type, and the count of each service offered at a specific mall located at 987 Pine Street. This query can help the customer know what services a specific mall provides. It joins the Mall, Mall_Service, and Service tables based on their respective IDs and then filters the results based on the mall's address. The results are then grouped by mall name and service type and ordered by the count of services in descending order.

```
SELECT Mall.Mall_Name, Service.Service_Type, COUNT(Service.Service_Type) AS Num_Services
FROM Mall
JOIN Mall_Service ON Mall.Mall_ID = Mall_Service.Mall_ID
JOIN Service ON Mall_Service.Service_ID = Service.Service_ID
WHERE Mall.Address = '987 Pine Street'
GROUP BY Mall.Mall_Name, Service.Service_Type
ORDER BY Num_Services DESC
```

MALL_NAME	SERVICE_TYPE	NUM_SERVICES
Mall E	Security Office	1
Mall E	Restaurant	1
Mall E	Play Area	1

2 - This query retrieves the store names and the corresponding count of items, specifically focusing on the item 'Eyeliner'. This query can help the customer search for a store that sells a specific item they want. It combines information from the Store, Mall_Store, and Store_Item tables. The Store and Mall_Store tables are joined to associate each store with its mall, and the Store_Item table is left-joined to include stores even if they don't have the specified item. The results are filtered to include only stores that sell the item ‘Eyeliner' , and then the counts of the items are calculated for each store. Finally, the results are grouped by store name and ordered alphabetically, providing a comprehensive overview of the number of 'Eyeliner' items sold by each store within the specified malls.

```
SELECT S.Store_Name, COUNT(SI.Item) AS Number_Of_Items
FROM Store S
JOIN Mall_Store MS ON S.Store_ID = MS.Store_ID
LEFT JOIN Store_Item SI ON S.Store_ID = SI.Store_ID
WHERE SI.Item = 'Eyeliner'
GROUP BY S.Store_Name
ORDER BY S.Store_Name;
```

STORE_NAME	NUMBER_OF_ITEMS
BeautyBoutique	1

3 - This query retrieves the total number of stores in each mall, by joining the Mall and Mall_Store tables and grouping the results by Mall_Name. This query can help mall mangers know if number of stores meet the number they desire, so that if it does not match a specific predefined number for example, they open more stores in the mall. The COUNT(Store_ID) function is used to count the number of stores in each mall. The results are then ordered in descending order based on the total number of stores.

```
SELECT Mall_Name, COUNT(Store_ID) AS Total_Stores
FROM Mall
JOIN Mall_Store ON Mall.Mall_ID = Mall_Store.Mall_ID
GROUP BY Mall_Name
ORDER BY Total_Stores DESC;
```

MALL_NAME	TOTAL_STORES
Mall E	3
Mall A	2
Mall C	2
Mall D	2
Mall B	1

4 - This query retrieves the first name, last name, mall name, and total visits of customers who have visited a mall of id 1. This query can help determine the favorite mall of the customer to make it a suggestion every time they use the application. It joins the Customer, Customer_Visit, and Mall tables using their respective IDs and then groups the results by customer first name, last name, and mall name. The results are then ordered by the total visits in descending order.

```
SELECT Customer.FirstName, Customer.LastName, Mall.Mall_Name, COUNT(Customer_Visit.Customer_ID) AS Total_Visits
FROM Customer
JOIN Customer_Visit ON Customer.Customer_ID = Customer_Visit.Customer_ID
JOIN Mall ON Customer_Visit.Mall_ID = Mall.Mall_ID
WHERE Mall.Mall_ID = 1
GROUP BY Customer.FirstName, Customer.LastName, Mall.Mall_Name
ORDER BY Total_Visits DESC;
```

FIRSTNAME	LASTNAME	MALL_NAME	TOTAL_VISITS
John	Doe	Mall A	1

Procedures:

PARAMETER based SELECT QUERY stored procedure:

The procedure "GetCustomerRecords" is designed to retrieve customer records based on the provided customer ID. Offering a simple and effective means of accessing customer information from the database.

```
CREATE OR REPLACE PROCEDURE GetCustomerRecords(
    ID IN Customer.Customer_ID%TYPE)
AS
    v_FirstName Customer.FirstName%TYPE;
    v_LastName Customer.LastName%TYPE;
    v_Email Customer.Email%TYPE;
    v_Password Customer.Password%TYPE;
BEGIN
    SELECT FirstName, LastName, Email, Password
    INTO v_FirstName, v_LastName, v_Email, v_Password
    FROM Customer
    WHERE Customer_ID = ID;

    DBMS_OUTPUT.PUT_LINE('First Name: ' || v_FirstName);
    DBMS_OUTPUT.PUT_LINE('Last Name: ' || v_LastName);
    DBMS_OUTPUT.PUT_LINE('Email: ' || v_Email);
    DBMS_OUTPUT.PUT_LINE('Password: ' || v_Password);
END;

```

```
Procedure created.
```

The result when the procedure is executed:

```
EXEC GetCustomerRecords(1);
```

```
Statement processed.
First Name: John
Last Name: Doe
Email: john.doe@example.com
Password: password123
```

UPDATE query based stored procedure:

The procedure, "UpdateMallContactNumber", takes two parameters - MallID and NewContactNumber. It then updates the "ContactNumber" for the mall specified by the MallID and commits the transaction. Additionally, exception handling is included to handle scenarios such as the mall not being found.

```
CREATE OR REPLACE PROCEDURE UpdateMallContactNumber(
    MallID IN Mall.Mall_ID%TYPE,
    NewContactNumber IN Mall.Contact_Number%TYPE)
AS
BEGIN
    UPDATE Mall
    SET Contact_Number = NewContactNumber
    WHERE Mall_ID = MallID;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Contact number updated successfully for Mall ID: ' || MallID);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No Mall found with ID: ' || MallID);

END;

```

```
Procedure created.
```

The result when the procedure is executed:

```
SELECT Mall_ID, Contact_Number
FROM Mall
WHERE Mall_ID = 2;

EXEC UpdateMallContactNumber (2, '666-777-8888');

SELECT Mall_ID, Contact_Number
FROM Mall
WHERE Mall_ID = 2;
```

MALL_ID	CONTACT_NUMBER
2	987-654-3210

Download CSV

```
Statement processed.
Contact number updated successfully for Mall ID: 2
```

MALL_ID	CONTACT_NUMBER
2	666-777-8888

Note: all of the SQL statements will be included in a txt file.