Department of Computer Science and Engineering
National Institute of Technology Warangal

DBMS PROJECT

Amusement Park Management System

BY :

Anjaneyulu Baikani

ABOUT:

Amusement Park Management System (APMS) is a centralized software platform designed to automate and streamline operations in theme parks. It integrates key functions such as ticketing, ride management, customer data tracking, point-of-sale (POS) systems, and workforce coordination into a unified digital ecosystem. By replacing manual processes with real-time data management, the system ensures efficient handling of visitor flow, safety compliance, revenue tracking, and maintenance scheduling. The APMS acts as the backbone of park operations, enabling administrators to monitor and optimize every aspect of the business from a single interface.

Amusement parks require an APMS to enhance efficiency, safety, and profitability. Manual management of tickets, ride queues, food sales, and employee schedules is error-prone and inefficient, especially in large parks with thousands of daily visitors. The system eliminates bottlenecks by automating ticketing, ensuring compliance with safety regulations through maintenance logs. Additionally, it boosts revenue by tracking spending patterns, enabling dynamic pricing, and facilitating targeted promotions. Without an APMS, parks risk operational chaos, lost revenue opportunities, and compromised guest satisfaction.

The APMS operates through interconnected modules that communicate in real time:

1. **Ride Management**: Sensors monitor ride capacity and safety inspections, flagging maintenance needs.
2. **POS & Retail Systems**: Cashless payments track food, merchandise, and photo sales per visitor.
3. **Analytics Dashboard**: Aggregates data on attendance, revenue peaks, and ride popularity for strategic decisions.
4. **Employee Management**: Assigns staff to rides, food outlets, or maintenance tasks based on real-time demand

The APMS enhances parks in **Improved Guest Experience, Operational Efficiency**, **Safety & Compliance** ,and **Data-Driven Growth.**

By integrating these functions, the system ensures smoother operations, higher visitor satisfaction, and increased profitability.

ENTITIES USED:

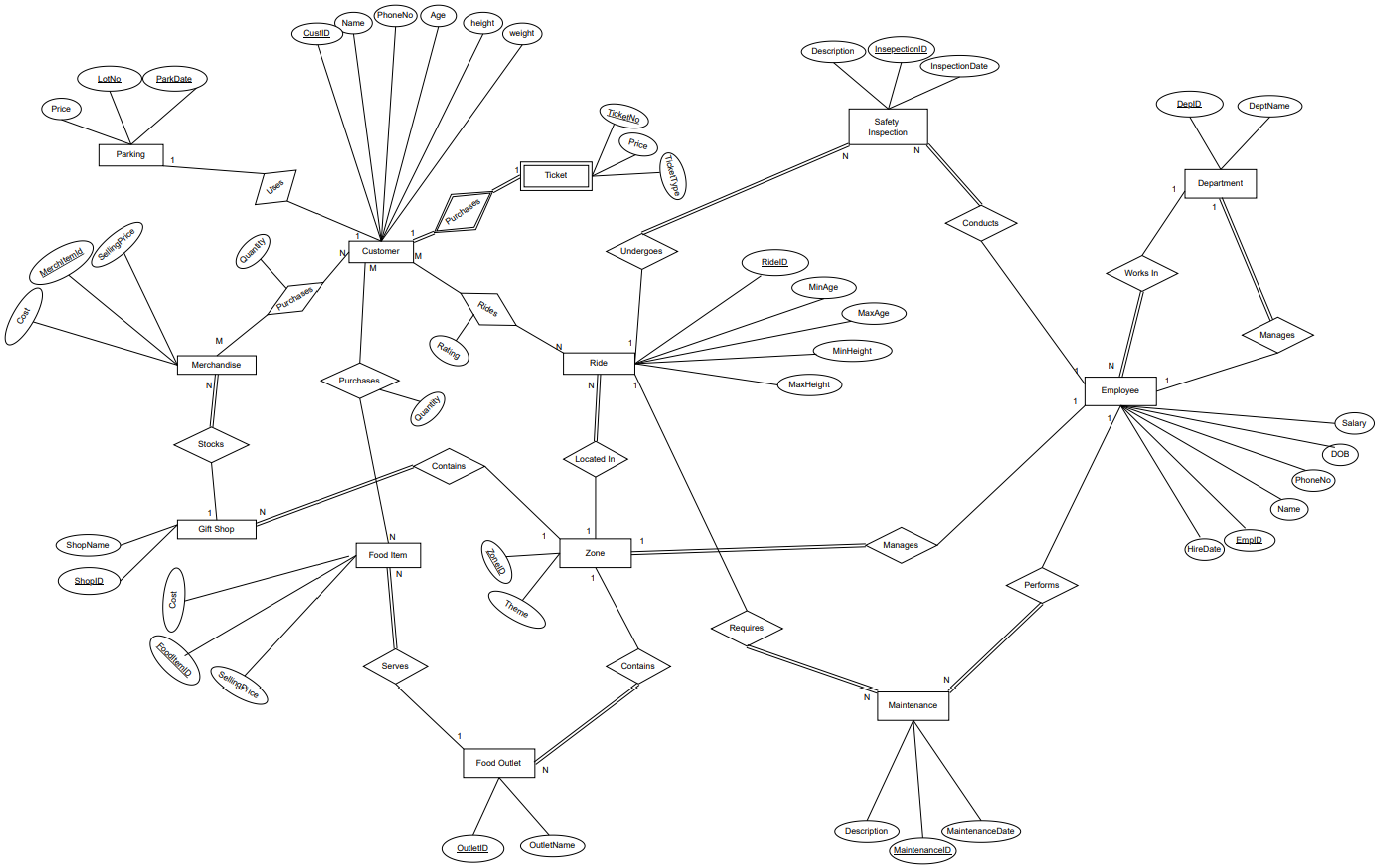
1. **Customer** - Represents park visitors who use services
2. **Ticket** - Represents admission passes purchased by customers
3. **Parking** - Represents vehicle parking used by customers
4. **Zone** - Represents themed areas containing attractions
5. **Ride** - Represents park attractions available to customers
6. **SafetyInspection** - Represents safety checks performed on rides
7. **Department** - Represents staff organizational units
8. **Employee** - Represents park workers maintaining operations
9. **FoodOutlet** - Represents dining locations in zones
10. **FoodItem** - Represents menu items sold at outlets
11. **GiftShop** - Represents retail stores in zones
12. **Merchandise** - Represents products sold in shops
13. **Maintenance** - Represents ride repair records

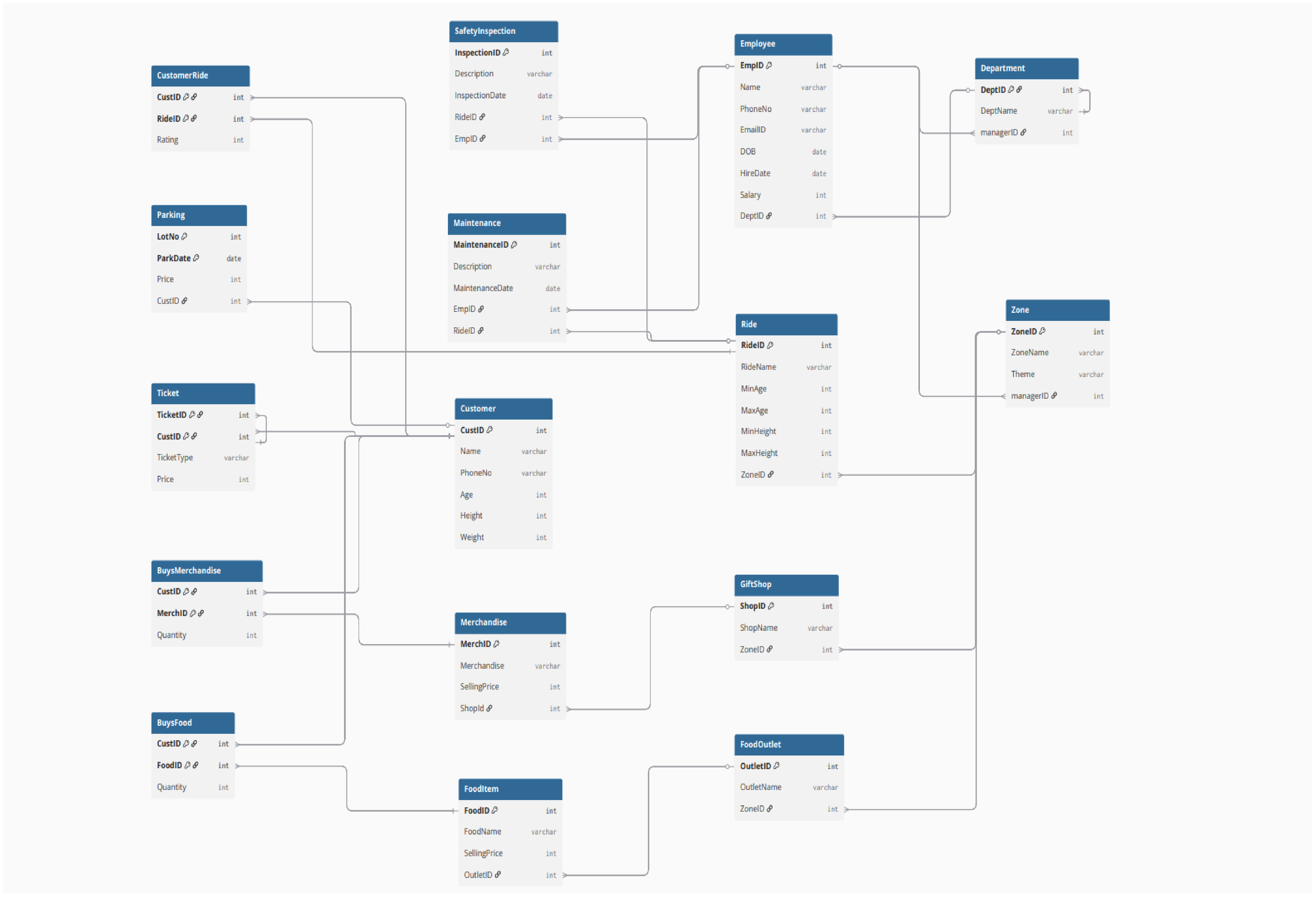
RELATIONSHIPS:

1. One Customer can purchase One Tickets (One-to-One)
2. One Customer can use One Parking slots (One-to-One)
3. One Zone can contain multiple Rides (One-to-Many)
4. One Ride requires multiple SafetyInspections (One-to-Many)
5. One Department employs multiple Employees (One-to-Many)
6. One Customer can ride multiple Rides (Many-to-Many via CustomerRide)
7. One Zone can have multiple FoodOutlets (One-to-Many)
8. One FoodOutlet offers multiple FoodItems (One-to-Many)
9. One Customer can buy multiple FoodItems (Many-to-Many via BuysFood)
10. One Zone can have multiple GiftShops (One-to-Many)

11. One GiftShop stocks multiple Merchandise items (One-to-Many)
12. One Customer can buy multiple Merchandise items (Many-to-Many via BuysMerchandise)
13. One Ride needs multiple Maintenance records (One-to-Many)
14. One Employee can perform multiple Maintenance tasks (One-to-Many)
15. One Employee can conduct multiple SafetyInspections (One-to-Many)

Amusement Park Managemet Systems




SafetyInspection
InspectionID int
Description varchar
InspectionDate date
RideID int
EmpID int
Employee
EmpID int
Name varchar
PhoneNo varchar
EmailID varchar
DOB date
HireDate date
Salary int
DeptID int
Department
DeptID int
DeptName varchar
managerID int
CustomerRide
CustID int
RideID int
Rating int
Parking
LotNo int
ParkDate date
Price int
CustID int
Maintenance
MaintenanceID int
Description varchar
MaintenanceDate date
EmpID int
RideID int
Ride
RideID int
RideName varchar
MinAge int
MaxAge int
MinHeight int
MaxHeight int
ZoneID int
Zone
ZoneID int
ZoneName varchar
Theme varchar
managerID int
Ticket
TicketID int
CustID int
TicketType varchar
Price int
Customer
CustID int
Name varchar
PhoneNo varchar
Age int
Height int
Weight int
BuysMerchandise
CustID int
MerchID int
Quantity int
Merchandise
MerchID int
Merchandise varchar
SellingPrice int
ShopId int
GiftShop
ShopID int
ShopName varchar
ZoneID int
BuysFood
CustID int
FoodID int
Quantity int
FoodItem
FoodID int
FoodName varchar
SellingPrice int
OutletID int
FoodOutlet
OutletID int
OutletName varchar
ZoneID int

TABLES:

1)Customer:

Attribute	Data Type	Constraints
CUST_ID	NUMERIC	Not null, Primary key
NAME	VARCHAR	Not null
PHONE_NO	VARCHAR	
AGE	NUMERIC	
HEIGHT	NUMERIC	
WEIGHT	NUMERIC	

2)Ticket:

Attribute	Data Type	Constraints
TICKET_ID	NUMERIC	Not null, Primary key
CUST_ID	NUMERIC	Not null, Foreign Key (Customer)
TICKET_TYPE	VARCHAR	Not null
PRICE	NUMERIC	Not null

3)Parking:

Attribute	Data Type	Constraints
LOT_NO	NUMERIC	Not null, Primary key
PARK_DATE	DATE	Not null
PRICE	NUMERIC	Not null
CUST_ID	NUMERIC	Not null, Foreign Key (Customer)

4)Zone:

Attribute	**Data Type**	**Constraints**
ZONE_ID	NUMERIC	Not null, Primary key
ZONE_NAME	VARCHAR	Not null
THEME	VARCHAR	
MANAGER_ID	NUMERIC	Foreign Key (Employee)

5)Ride:

Attribute	**Data Type**	**Constraints**
RIDE_ID	NUMERIC	Not null, Primary key
RIDE_NAME	VARCHAR	Not null
MIN_AGE	NUMERIC	
MAX_AGE	NUMERIC	
MIN_HEIGHT	NUMERIC	
MAX_HEIGHT	NUMERIC	
ZONE_ID	NUMERIC	Not null, Foreign Key (Zone)

6)Safety_Insepection:

Attribute	**Data Type**	**Constraints**
INSPECTION_ID	NUMERIC	Not null, Primary key
DESCRIPTION	VARCHAR	Not null
INSPECTION_DATE	DATE	Not null
RIDE_ID	NUMERIC	Not null, Foreign Key (Ride)
EMP_ID	NUMERIC	Not null, Foreign Key (Employee)

7)Department:

Attribute	**Data Type**	**Constraints**
DEPT_ID	NUMERIC	Not null, Primary key
DEPT_NAME	VARCHAR	Not null
MANAGER_ID	NUMERIC	Foreign Key (Employee)

8) EMPLOYEE:

Attribute	**Data Type**	**Constraints**
EMP_ID	NUMERIC	Not null, Primary key
NAME	VARCHAR	Not null
PHONE_NO	VARCHAR	
EMAIL_ID	VARCHAR	
DOB	DATE	
HIRE_DATE	DATE	
SALARY	NUMERIC	
DEPT_ID	NUMERIC	Not null, Foreign Key (Department)

9) CUSTOMER_RIDE:

Attribute	Data Type	Constraints
CUST_ID	NUMERIC	Not null, Foreign Key (Customer)
RIDE_ID	NUMERIC	Not null, Foreign Key (Ride)
RATING	NUMERIC	
PRIMARY KEY (CUST_ID, RIDE_ID)		

10) FOOD_OUTLET:

Attribute	Data Type	Constraints
OUTLET_ID	NUMERIC	Not null, Primary key
OUTLET_NAME	VARCHAR	Not null
ZONE_ID	NUMERIC	Not null, Foreign Key (Zone)

11) FOOD_ITEM:

Attribute	Data Type	Constraints
FOOD_ID	NUMERIC	Not null, Primary key
FOOD_NAME	VARCHAR	Not null
SELLING_PRICE	NUMERIC	Not null
OUTLET_ID	NUMERIC	Not null, Foreign Key (FoodOutlet)

12) BUYS_FOOD:

Attribute	Data Type	Constraints
CUST_ID	NUMERIC	Not null, Foreign Key (Customer)
FOOD_ID	NUMERIC	Not null, Foreign Key (FoodItem)
QUANTITY	NUMERIC	Not null
PRIMARY KEY (CUST_ID, FOOD_ID)		

13) GIFT_SHOP:

Attribute	Data Type	Constraints
SHOP_ID	NUMERIC	Not null, Primary key
SHOP_NAME	VARCHAR	Not null
ZONE_ID	NUMERIC	Not null, Foreign Key (Zone)

14) MERCHANDISE:

Attribute	Data Type	Constraints
MERCH_ID	NUMERIC	Not null, Primary key
MERCHANDISE_NAME	VARCHAR	Not null
SELLING_PRICE	NUMERIC	Not null
SHOP_ID	NUMERIC	Not null, Foreign Key (GiftShop)

15) BUYS_MERCHANDISE:

Attribute	Data Type	Constraints
CUST_ID	NUMERIC	Not null, Foreign Key (Customer)
MERCH_ID	NUMERIC	Not null, Foreign Key (Merchandise)
QUANTITY	NUMERIC	Not null
PRIMARY KEY (CUST_ID, MERCH_ID)		

16) MAINTENANCE:

Attribute	Data Type	Constraints
MAINTENANCE_ID	NUMERIC	Not null, Primary key
DESCRIPTION	VARCHAR	Not null
MAINTENANCE_DATE	DATE	Not null
EMP_ID	NUMERIC	Not null, Foreign Key (Employee)
RIDE_ID	NUMERIC	Not null, Foreign Key (Ride)

NORMALIZAITON:

1) CUSTOMER

Cust_ID → (Name, Phone_No, Age, Height, Weight)

Primary Key = Cust_ID
Prime Attributes = Cust_ID
Non-Prime Attributes = Name, Phone_No, Age, Height, Weight

- No multivalued/composite attributes → **1NF**
- No partial dependencies (single-column PK) → **2NF**
- No transitive dependencies → **3NF**
- All dependencies come from candidate key → **BCNF**

2) TICKET

Ticket_ID → (Cust_ID, Ticket_Type, Price)

Primary Key = Ticket_ID
Prime Attributes = Ticket_ID
Non-Prime Attributes = Cust_ID, Ticket_Type, Price

- No multivalued/composite attributes → **1NF**
- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

3) PARKING

Lot_No → (Park_Date, Price, Cust_ID)

Primary Key = Lot_No
Prime Attributes = Lot_No
Non-Prime Attributes = Park_Date, Price, Cust_ID

- No multivalued attributes → **1NF**

- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

4) <u>ZONE</u>

Zone_ID → (Zone_Name, Theme, Manager_ID)

Manager_ID->Zone_ID

Primary Key = Zone_ID
Prime Attributes = Zone_ID,Manager_ID
Non-Prime Attributes = Zone_Name, Theme

- No multivalued attributes → **1NF**
- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

5) <u>RIDE</u>

Ride_ID → (Ride_Name, Min_Age, Max_Age, Min_Height, Max_Height, Zone_ID)

Primary Key = Ride_ID
Prime Attributes = Ride_ID
Non-Prime Attributes = Ride_Name, Min_Age, Max_Age, Min_Height, Max_Height, Zone_ID

- No multivalued attributes → **1NF**
- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

6) SAFETY_INSPECTION

Inspection_ID → (Description, Inspection_Date, Ride_ID, Emp_ID)

Primary Key = Inspection_ID
Prime Attributes = Inspection_ID
Non-Prime Attributes = Description, Inspection_Date, Ride_ID, Emp_ID

- No multivalued attributes → **1NF**
- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

7) DEPARTMENT

Dept_ID → (Dept_Name, Manager_ID)

Manager_ID->Dept_ID

Primary Key = Dept_ID
Prime Attributes = Dept_ID, Manager_ID
Non-Prime Attributes = Dept_Name

- No multivalued attributes → **1NF**
- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

8) EMPLOYEE

Emp_ID → (Name, Phone_No, Email_ID, DOB, Hire_Date, Salary, Dept_ID)

Primary Key = Emp_ID
Prime Attributes = Emp_ID
Non-Prime Attributes = Name, Phone_No, Email_ID, DOB, Hire_Date, Salary, Dept_ID

- No multivalued attributes → **1NF**

- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

9) <u>CUSTOMER_RIDE</u>

(Cust_ID, Ride_ID) → Rating

Composite Primary Key = (Cust_ID, Ride_ID)
Prime Attributes = Cust_ID, Ride_ID
Non-Prime Attributes = Rating

- No multivalued attributes → **1NF**
- No partial dependencies (all non-prime depend on full PK) → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

10) <u>FOOD_OUTLET</u>

Outlet_ID → (Outlet_Name, Zone_ID)

Primary Key = Outlet_ID
Prime Attributes = Outlet_ID
Non-Prime Attributes = Outlet_Name, Zone_ID

- No multivalued attributes → **1NF**
- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

11) <u>FOOD_ITEM</u>

Food_ID → (Food_Name, Selling_Price, Outlet_ID)

Primary Key = Food_ID
Prime Attributes = Food_ID
Non-Prime Attributes = Food_Name, Selling_Price, Outlet_ID

- No multivalued attributes → **1NF**
- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

12) BUYS_FOOD

(Cust_ID, Food_ID) → Quantity

Composite Primary Key = (Cust_ID, Food_ID)
Prime Attributes = Cust_ID, Food_ID
Non-Prime Attributes = Quantity

- No multivalued attributes → **1NF**
- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

13) GIFT_SHOP

Shop_ID → (Shop_Name, Zone_ID)

Primary Key = Shop_ID
Prime Attributes = Shop_ID
Non-Prime Attributes = Shop_Name, Zone_ID

- No multivalued attributes → **1NF**
- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

14) MERCHANDISE

Merch_ID → (Merchandise_Name, Selling_Price, Shop_ID)

Primary Key = Merch_ID
Prime Attributes = Merch_ID
Non-Prime Attributes = Merchandise_Name, Selling_Price, Shop_ID

- No multivalued attributes → **1NF**
- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

15) BUYS_MERCHANDISE

(Cust_ID, Merch_ID) → Quantity

Composite Primary Key = (Cust_ID, Merch_ID)
Prime Attributes = Cust_ID, Merch_ID
Non-Prime Attributes = Quantity

- No multivalued attributes → **1NF**
- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

16) MAINTENANCE

Maintenance_ID → (Description, Maintenance_Date, Emp_ID, Ride_ID)

Primary Key = Maintenance_ID
Prime Attributes = Maintenance_ID
Non-Prime Attributes = Description, Maintenance_Date, Emp_ID, Ride_ID

- No multivalued attributes → **1NF**
- No partial dependencies → **2NF**
- No transitive dependencies → **3NF**
- All dependencies from candidate key → **BCNF**

CREATING TABLES:

1)DEPARTMENT:

```
CREATE TABLE Department (
   DeptID INT PRIMARY KEY,
   DeptName VARCHAR(50) NOT NULL,
   managerID INT
);
```

INSERTING VALUES:

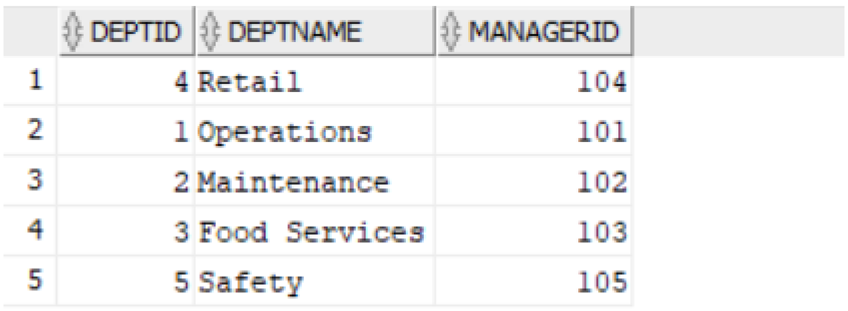
```
INSERT INTO Department VALUES (1, 'Operations', NULL);
INSERT INTO Department VALUES (2, 'Maintenance', NULL);
INSERT INTO Department VALUES (3, 'Food Services', NULL);
INSERT INTO Department VALUES (4, 'Retail', NULL);
INSERT INTO Department VALUES (5, 'Safety', NULL);
```

ADD FOREIGN KEY TO DEPARTMENT TABLE AFTER CREATING EMPLOYEE TABLE:

```
ALTER TABLE Department ADD CONSTRAINT fk_dept_manager
   FOREIGN KEY (managerID) REFERENCES Employee(EmpID) DEFERRABLE
INITIALLY DEFERRED;
UPDATE Department SET managerID = 101 WHERE DeptID = 1;
UPDATE Department SET managerID = 102 WHERE DeptID = 2;
UPDATE Department SET managerID = 103 WHERE DeptID = 3;
UPDATE Department SET managerID = 104 WHERE DeptID = 4;
```

UPDATE Department SET managerID = 105 WHERE DeptID = 5;

OUTPUT:



	DEPTID	DEPTNAME	MANAGERID
1	4	Retail	104
2	1	Operations	101
3	2	Maintenance	102
4	3	Food Services	103
5	5	Safety	105

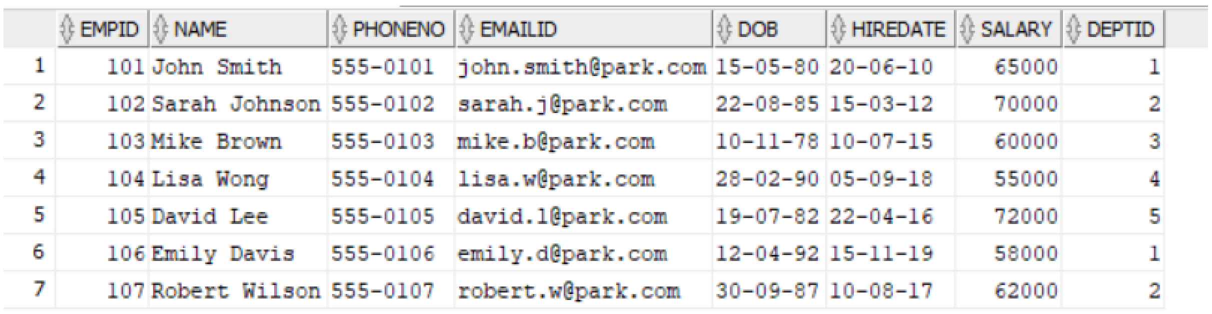
2)EMPLOYEE:

```
CREATE TABLE Employee (
   EmpID INT PRIMARY KEY,
   Name VARCHAR(100) NOT NULL,
   PhoneNo VARCHAR(20),
   EmailID VARCHAR(100),
   DOB DATE,
   HireDate DATE,
   Salary INT,
   DeptID INT,
   CONSTRAINT fk_emp_dept FOREIGN KEY (DeptID) REFERENCES
Department(DeptID)
);
```

INSERTING VALUES:

```
INSERT INTO Employee VALUES (101, 'John Smith', '555-0101',
'john.smith@park.com', TO_DATE('1980-05-15', 'YYYY-MM-DD'),
TO_DATE('2010-06-20', 'YYYY-MM-DD'), 65000, 1);

INSERT INTO Employee VALUES (102, 'Sarah Johnson', '555-0102',
'sarah.j@park.com', TO_DATE('1985-08-22', 'YYYY-MM-DD'), TO_DATE('2012-
03-15', 'YYYY-MM-DD'), 70000, 2);

INSERT INTO Employee VALUES (103, 'Mike Brown', '555-0103',
'mike.b@park.com', TO_DATE('1978-11-10', 'YYYY-MM-DD'), TO_DATE('2015-
07-10', 'YYYY-MM-DD'), 60000, 3);

INSERT INTO Employee VALUES (104, 'Lisa Wong', '555-0104',
'lisa.w@park.com', TO_DATE('1990-02-28', 'YYYY-MM-DD'), TO_DATE('2018-09-
05', 'YYYY-MM-DD'), 55000, 4);

INSERT INTO Employee VALUES (105, 'David Lee', '555-0105',
'david.l@park.com', TO_DATE('1982-07-19', 'YYYY-MM-DD'), TO_DATE('2016-
04-22', 'YYYY-MM-DD'), 72000, 5);

INSERT INTO Employee VALUES (106, 'Emily Davis', '555-0106',
'emily.d@park.com', TO_DATE('1992-04-12', 'YYYY-MM-DD'), TO_DATE('2019-
11-15', 'YYYY-MM-DD'), 58000, 1);

INSERT INTO Employee VALUES (107, 'Robert Wilson', '555-0107',
'robert.w@park.com', TO_DATE('1987-09-30', 'YYYY-MM-DD'), TO_DATE('2017-
08-10', 'YYYY-MM-DD'), 62000, 2);
```

OUTPUT:



```
   EMPID NAME          PHONENO  EMAILID             DOB      HIREDATE SALARY DEPTID
1    101 John Smith    555-0101 john.smith@park.com 15-05-80 20-06-10  65000      1
2    102 Sarah Johnson 555-0102 sarah.j@park.com    22-08-85 15-03-12  70000      2
3    103 Mike Brown    555-0103 mike.b@park.com     10-11-78 10-07-15  60000      3
4    104 Lisa Wong     555-0104 lisa.w@park.com     28-02-90 05-09-18  55000      4
5    105 David Lee     555-0105 david.l@park.com    19-07-82 22-04-16  72000      5
6    106 Emily Davis   555-0106 emily.d@park.com    12-04-92 15-11-19  58000      1
7    107 Robert Wilson 555-0107 robert.w@park.com   30-09-87 10-08-17  62000      2
```

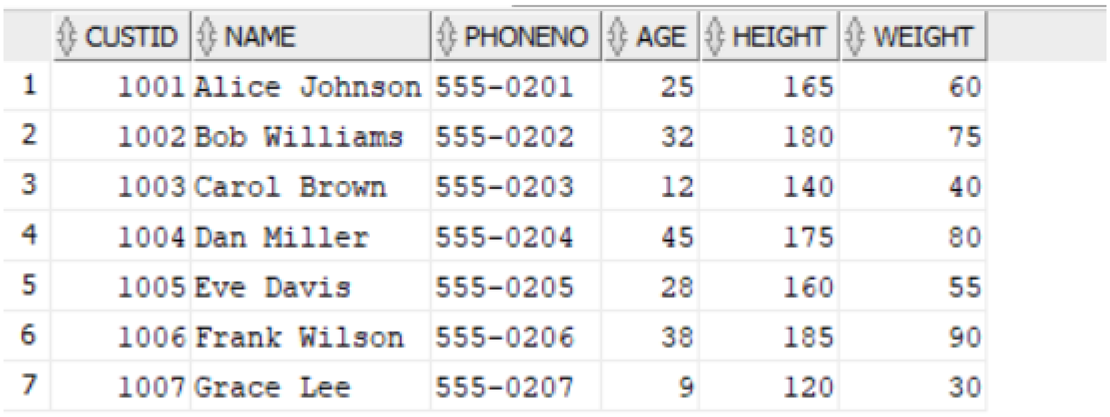
3)CUSTOMER:

```
CREATE TABLE Customer (
   CustID INT PRIMARY KEY,
   Name VARCHAR(100) NOT NULL,
   PhoneNo VARCHAR(20),
   Age INT,
   Height INT,
   Weight INT
);
```

INSERTING VALUES:

```
INSERT INTO Customer VALUES (1001, 'Alice Johnson', '555-0201', 25, 165, 60);
INSERT INTO Customer VALUES (1002, 'Bob Williams', '555-0202', 32, 180, 75);
INSERT INTO Customer VALUES (1003, 'Carol Brown', '555-0203', 12, 140, 40);
INSERT INTO Customer VALUES (1004, 'Dan Miller', '555-0204', 45, 175, 80);
INSERT INTO Customer VALUES (1005, 'Eve Davis', '555-0205', 28, 160, 55);
INSERT INTO Customer VALUES (1006, 'Frank Wilson', '555-0206', 38, 185, 90);
INSERT INTO Customer VALUES (1007, 'Grace Lee', '555-0207', 9, 120, 30);
```

OUTPUT:



	CUSTID	NAME	PHONENO	AGE	HEIGHT	WEIGHT
1	1001	Alice Johnson	555-0201	25	165	60
2	1002	Bob Williams	555-0202	32	180	75
3	1003	Carol Brown	555-0203	12	140	40
4	1004	Dan Miller	555-0204	45	175	80
5	1005	Eve Davis	555-0205	28	160	55
6	1006	Frank Wilson	555-0206	38	185	90
7	1007	Grace Lee	555-0207	9	120	30

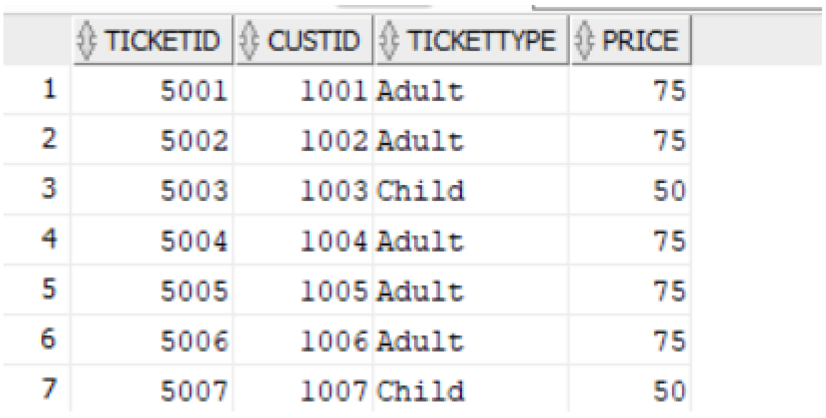
4)TICKET:

```
CREATE TABLE Ticket (
   TicketID INT PRIMARY KEY,
   CustID INT NOT NULL,
   TicketType VARCHAR(50),
   Price INT,
   CONSTRAINT fk_ticket_customer FOREIGN KEY (CustID) REFERENCES
Customer(CustID)
);
```

INSERTING VALUES:

```
INSERT INTO Ticket VALUES (5001, 1001, 'Adult', 75);
INSERT INTO Ticket VALUES (5002, 1002, 'Adult', 75);
INSERT INTO Ticket VALUES (5003, 1003, 'Child', 50);
INSERT INTO Ticket VALUES (5004, 1004, 'Adult', 75);
INSERT INTO Ticket VALUES (5005, 1005, 'Adult', 75);
INSERT INTO Ticket VALUES (5006, 1006, 'Adult', 75);
INSERT INTO Ticket VALUES (5007, 1007, 'Child', 50);
```

OUTPUT:



	TICKETID	CUSTID	TICKETTYPE	PRICE
1	5001	1001	Adult	75
2	5002	1002	Adult	75
3	5003	1003	Child	50
4	5004	1004	Adult	75
5	5005	1005	Adult	75
6	5006	1006	Adult	75
7	5007	1007	Child	50

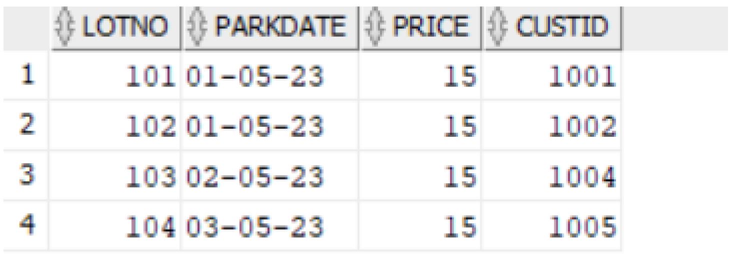
5)PARKING:

```
CREATE TABLE Parking (
   LotNo INT PRIMARY KEY,
   ParkDate DATE,
   Price INT,
   CustID INT,
   CONSTRAINT fk_parking_customer FOREIGN KEY (CustID) REFERENCES
Customer(CustID)
);
```

INSERTING VALUES:

```
INSERT INTO Parking VALUES (101, TO_DATE('2023-05-01', 'YYYY-MM-DD'), 15,
1001);
INSERT INTO Parking VALUES (102, TO_DATE('2023-05-01', 'YYYY-MM-DD'), 15,
1002);
INSERT INTO Parking VALUES (103, TO_DATE('2023-05-02', 'YYYY-MM-DD'), 15,
1004);
INSERT INTO Parking VALUES (104, TO_DATE('2023-05-03', 'YYYY-MM-DD'), 15,
1005);
```

OUTPUT:



	LOTNO	PARKDATE	PRICE	CUSTID
1	101	01-05-23	15	1001
2	102	01-05-23	15	1002
3	103	02-05-23	15	1004
4	104	03-05-23	15	1005

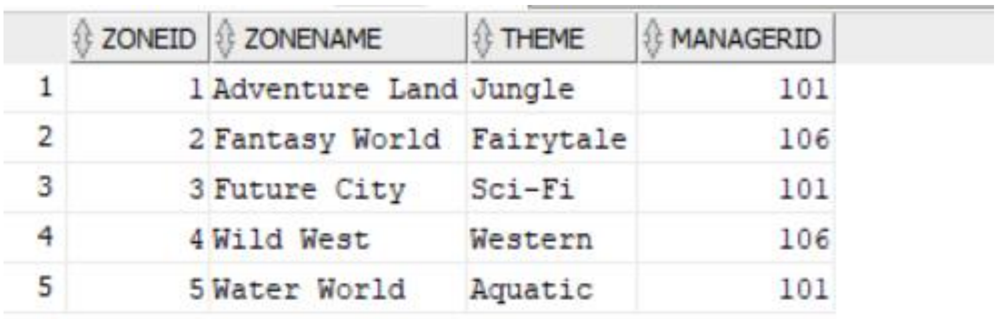
6)ZONE:

```
CREATE TABLE Zone (
    ZoneID INT PRIMARY KEY,
    ZoneName VARCHAR(50) NOT NULL,
    Theme VARCHAR(50),
    managerID INT,
    CONSTRAINT fk_zone_manager FOREIGN KEY (managerID) REFERENCES
Employee(EmpID)
);
```

INSERTING VALUES:

```
INSERT INTO Zone VALUES (1, 'Adventure Land', 'Jungle', 101);
INSERT INTO Zone VALUES (2, 'Fantasy World', 'Fairytale', 106);
INSERT INTO Zone VALUES (3, 'Future City', 'Sci-Fi', 101);
INSERT INTO Zone VALUES (4, 'Wild West', 'Western', 106);
INSERT INTO Zone VALUES (5, 'Water World', 'Aquatic', 101);
```

OUTPUT:



	ZONEID	ZONENAME	THEME	MANAGERID
1	1	Adventure Land	Jungle	101
2	2	Fantasy World	Fairytale	106
3	3	Future City	Sci-Fi	101
4	4	Wild West	Western	106
5	5	Water World	Aquatic	101

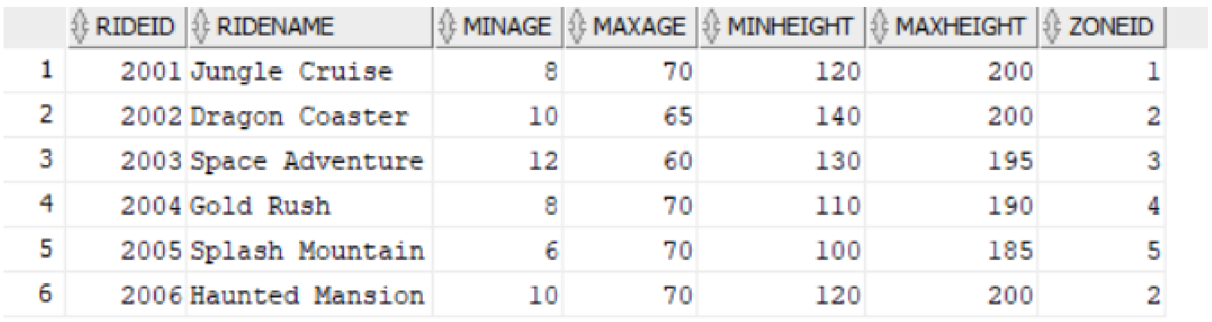
7)RIDE:

```
CREATE TABLE Ride (
   RideID INT PRIMARY KEY,
   RideName VARCHAR(100) NOT NULL,
   MinAge INT,
   MaxAge INT,
   MinHeight INT,
   MaxHeight INT,
   ZoneID INT,
   CONSTRAINT fk_ride_zone FOREIGN KEY (ZoneID) REFERENCES Zone(ZoneID)
);
```

INSERTING VALUES:

```
INSERT INTO Ride VALUES (2001, 'Jungle Cruise', 8, 70, 120, 200, 1);
INSERT INTO Ride VALUES (2002, 'Dragon Coaster', 10, 65, 140, 200, 2);
INSERT INTO Ride VALUES (2003, 'Space Adventure', 12, 60, 130, 195, 3);
INSERT INTO Ride VALUES (2004, 'Gold Rush', 8, 70, 110, 190, 4);
INSERT INTO Ride VALUES (2005, 'Splash Mountain', 6, 70, 100, 185, 5);
INSERT INTO Ride VALUES (2006, 'Haunted Mansion', 10, 70, 120, 200, 2);
```

OUTPUT:



	RIDEID	RIDENAME	MINAGE	MAXAGE	MINHEIGHT	MAXHEIGHT	ZONEID
1	2001	Jungle Cruise	8	70	120	200	1
2	2002	Dragon Coaster	10	65	140	200	2
3	2003	Space Adventure	12	60	130	195	3
4	2004	Gold Rush	8	70	110	190	4
5	2005	Splash Mountain	6	70	100	185	5
6	2006	Haunted Mansion	10	70	120	200	2

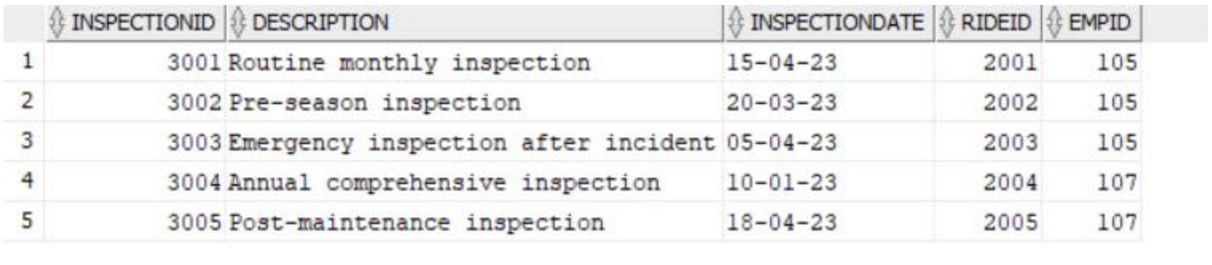
8)SAFETYINSPECTION:

```
CREATE TABLE SafetyInspection (
   InspectionID INT PRIMARY KEY,
   Description VARCHAR(200),
   InspectionDate DATE,
   RideID INT NOT NULL,
   EmpID INT NOT NULL,
   CONSTRAINT fk_inspection_ride FOREIGN KEY (RideID) REFERENCES Ride(RideID),
   CONSTRAINT fk_inspection_employee FOREIGN KEY (EmpID) REFERENCES Employee(EmpID)
);
```

INSERTING VALUES:

```
INSERT INTO SafetyInspection VALUES (3001, 'Routine monthly inspection', TO_DATE('2023-04-15', 'YYYY-MM-DD'), 2001, 105);
INSERT INTO SafetyInspection VALUES (3002, 'Pre-season inspection', TO_DATE('2023-03-20', 'YYYY-MM-DD'), 2002, 105);
INSERT INTO SafetyInspection VALUES (3003, 'Emergency inspection after incident', TO_DATE('2023-04-05', 'YYYY-MM-DD'), 2003, 105);
INSERT INTO SafetyInspection VALUES (3004, 'Annual comprehensive inspection', TO_DATE('2023-01-10', 'YYYY-MM-DD'), 2004, 107);
INSERT INTO SafetyInspection VALUES (3005, 'Post-maintenance inspection', TO_DATE('2023-04-18', 'YYYY-MM-DD'), 2005, 107);
```

OUTPUT:



	INSPECTIONID	DESCRIPTION	INSPECTIONDATE	RIDEID	EMPID
1	3001	Routine monthly inspection	15-04-23	2001	105
2	3002	Pre-season inspection	20-03-23	2002	105
3	3003	Emergency inspection after incident	05-04-23	2003	105
4	3004	Annual comprehensive inspection	10-01-23	2004	107
5	3005	Post-maintenance inspection	18-04-23	2005	107

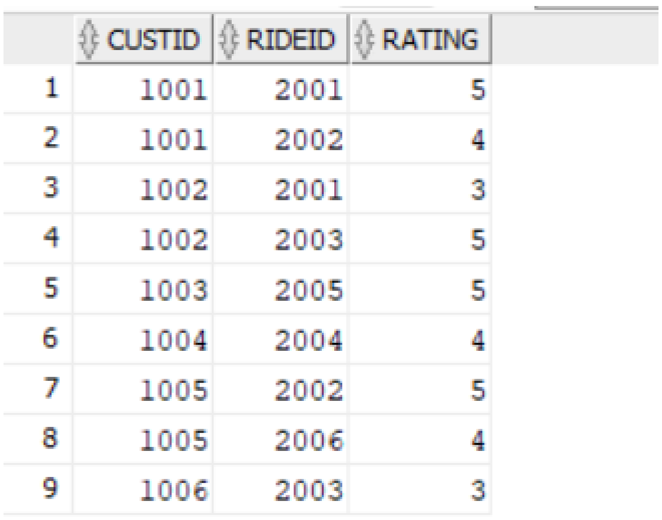
9)CUSTOMERRIDE:

```
CREATE TABLE CustomerRide (
   CustID INT,
   RideID INT,
   Rating INT CHECK (Rating BETWEEN 1 AND 5),
   PRIMARY KEY (CustID, RideID),
   CONSTRAINT fk_customerride_customer FOREIGN KEY (CustID) REFERENCES
Customer(CustID),
   CONSTRAINT fk_customerride_ride FOREIGN KEY (RideID) REFERENCES
Ride(RideID)
);
```

INSERTING VALUES:

```
INSERT INTO CustomerRide VALUES (1001, 2001, 5);
INSERT INTO CustomerRide VALUES (1001, 2002, 4);
INSERT INTO CustomerRide VALUES (1002, 2001, 3);
INSERT INTO CustomerRide VALUES (1002, 2003, 5);
INSERT INTO CustomerRide VALUES (1003, 2005, 5);
INSERT INTO CustomerRide VALUES (1004, 2004, 4);
INSERT INTO CustomerRide VALUES (1005, 2002, 5);
INSERT INTO CustomerRide VALUES (1005, 2006, 4);
INSERT INTO CustomerRide VALUES (1006, 2003, 3);
```

OUTPUT:



	CUSTID	RIDEID	RATING
1	1001	2001	5
2	1001	2002	4
3	1002	2001	3
4	1002	2003	5
5	1003	2005	5
6	1004	2004	4
7	1005	2002	5
8	1005	2006	4
9	1006	2003	3

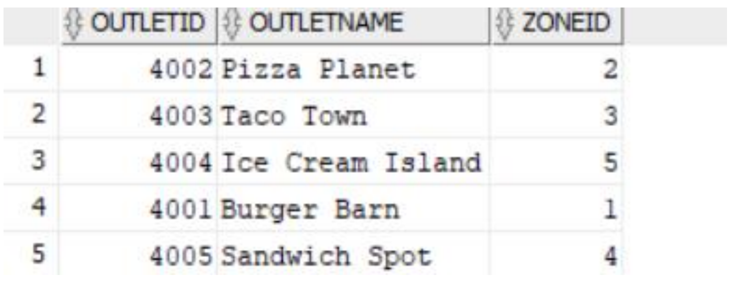
10)FOODOUTLET:

```
CREATE TABLE FoodOutlet (
   OutletID INT PRIMARY KEY,
   OutletName VARCHAR(100) NOT NULL,
   ZoneID INT,
   CONSTRAINT fk_foodoutlet_zone FOREIGN KEY (ZoneID) REFERENCES
Zone(ZoneID)
);
```

INSERTING VALUES:

```
INSERT INTO FoodOutlet VALUES (4001, 'Burger Barn', 1);
INSERT INTO FoodOutlet VALUES (4002, 'Pizza Planet', 2);
INSERT INTO FoodOutlet VALUES (4003, 'Taco Town', 3);
INSERT INTO FoodOutlet VALUES (4004, 'Ice Cream Island', 5);
INSERT INTO FoodOutlet VALUES (4005, 'Sandwich Spot', 4);
```

OUTPUT:



	OUTLETID	OUTLETNAME	ZONEID
1	4002	Pizza Planet	2
2	4003	Taco Town	3
3	4004	Ice Cream Island	5
4	4001	Burger Barn	1
5	4005	Sandwich Spot	4

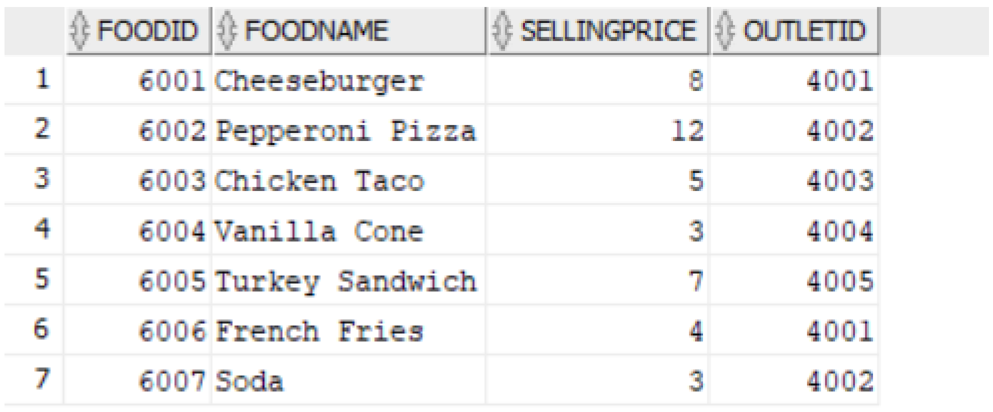
11)FOODITEM:

```
CREATE TABLE FoodItem (
   FoodID INT PRIMARY KEY,
   FoodName VARCHAR(100) NOT NULL,
   SellingPrice INT,
   OutletID INT,
   CONSTRAINT fk_fooditem_outlet FOREIGN KEY (OutletID) REFERENCES
FoodOutlet(OutletID)
);
```

INSERTING VALUES:

```
INSERT INTO FoodItem VALUES (6001, 'Cheeseburger', 8, 4001);
INSERT INTO FoodItem VALUES (6002, 'Pepperoni Pizza', 12, 4002);
INSERT INTO FoodItem VALUES (6003, 'Chicken Taco', 5, 4003);
INSERT INTO FoodItem VALUES (6004, 'Vanilla Cone', 3, 4004);
INSERT INTO FoodItem VALUES (6005, 'Turkey Sandwich', 7, 4005);
INSERT INTO FoodItem VALUES (6006, 'French Fries', 4, 4001);
INSERT INTO FoodItem VALUES (6007, 'Soda', 3, 4002);
```

OUTPUT:



	FOODID	FOODNAME	SELLINGPRICE	OUTLETID
1	6001	Cheeseburger	8	4001
2	6002	Pepperoni Pizza	12	4002
3	6003	Chicken Taco	5	4003
4	6004	Vanilla Cone	3	4004
5	6005	Turkey Sandwich	7	4005
6	6006	French Fries	4	4001
7	6007	Soda	3	4002

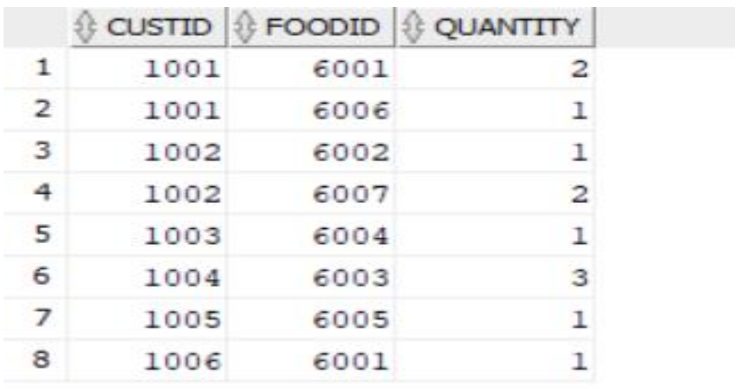
12)BUYSFOOD:

```
CREATE TABLE BuysFood (
   CustID INT,
   FoodID INT,
   Quantity INT,
   PRIMARY KEY (CustID, FoodID),
   CONSTRAINT fk_buysfood_customer FOREIGN KEY (CustID) REFERENCES
Customer(CustID),
   CONSTRAINT fk_buysfood_fooditem FOREIGN KEY (FoodID) REFERENCES
FoodItem(FoodID)
);
```

INSERTING VALUES:

```
INSERT INTO BuysFood VALUES (1001, 6001, 2);
INSERT INTO BuysFood VALUES (1001, 6006, 1);
INSERT INTO BuysFood VALUES (1002, 6002, 1);
INSERT INTO BuysFood VALUES (1002, 6007, 2);
INSERT INTO BuysFood VALUES (1003, 6004, 1);
INSERT INTO BuysFood VALUES (1004, 6003, 3);
INSERT INTO BuysFood VALUES (1005, 6005, 1);
INSERT INTO BuysFood VALUES (1006, 6001, 1);
```

OUTPUT:



	CUSTID	FOODID	QUANTITY
1	1001	6001	2
2	1001	6006	1
3	1002	6002	1
4	1002	6007	2
5	1003	6004	1
6	1004	6003	3
7	1005	6005	1
8	1006	6001	1

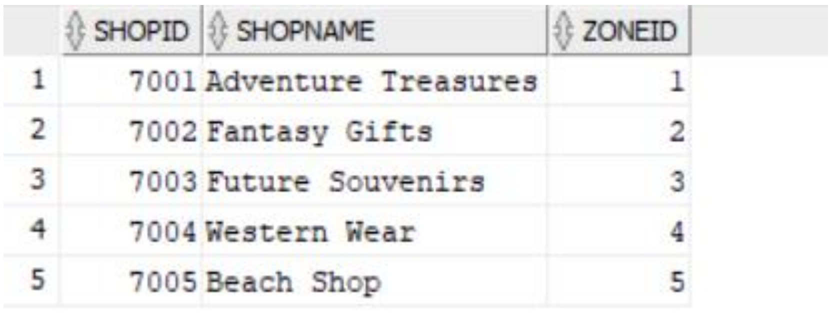
13)GIFTSHOP:

```
CREATE TABLE GiftShop (
   ShopID INT PRIMARY KEY,
   ShopName VARCHAR(100) NOT NULL,
   ZoneID INT,
   CONSTRAINT fk_giftshop_zone FOREIGN KEY (ZoneID) REFERENCES
Zone(ZoneID)
);
```

INSERTING VALUES:

```
INSERT INTO GiftShop VALUES (7001, 'Adventure Treasures', 1);
INSERT INTO GiftShop VALUES (7002, 'Fantasy Gifts', 2);
INSERT INTO GiftShop VALUES (7003, 'Future Souvenirs', 3);
INSERT INTO GiftShop VALUES (7004, 'Western Wear', 4);
INSERT INTO GiftShop VALUES (7005, 'Beach Shop', 5);
```

OUTPUT:



	SHOPID	SHOPNAME	ZONEID
1	7001	Adventure Treasures	1
2	7002	Fantasy Gifts	2
3	7003	Future Souvenirs	3
4	7004	Western Wear	4
5	7005	Beach Shop	5

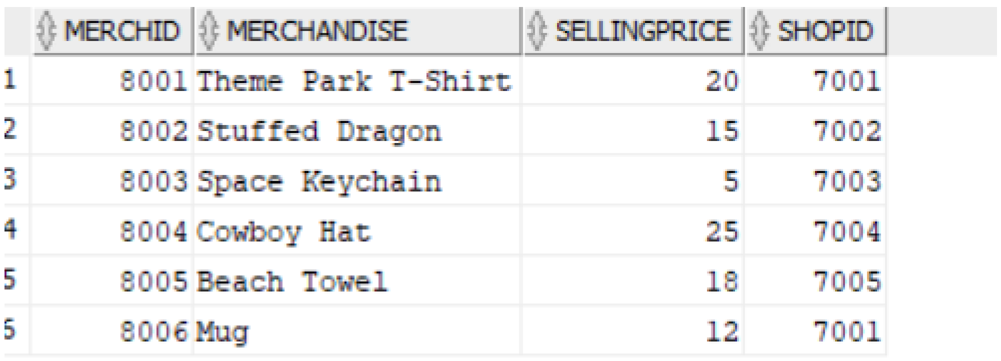
14)MERCHANDISE:

```
CREATE TABLE Merchandise (

   MerchID INT PRIMARY KEY,

   Merchandise VARCHAR(100) NOT NULL,

   SellingPrice INT,

   ShopID INT,

   CONSTRAINT fk_merchandise_shop FOREIGN KEY (ShopID) REFERENCES
GiftShop(ShopID)

);
```

INSERTING VALUES:

```
INSERT INTO Merchandise VALUES (8001, 'Theme Park T-Shirt', 20, 7001);

INSERT INTO Merchandise VALUES (8002, 'Stuffed Dragon', 15, 7002);

INSERT INTO Merchandise VALUES (8003, 'Space Keychain', 5, 7003);

INSERT INTO Merchandise VALUES (8004, 'Cowboy Hat', 25, 7004);

INSERT INTO Merchandise VALUES (8005, 'Beach Towel', 18, 7005);

INSERT INTO Merchandise VALUES (8006, 'Mug', 12, 7001);
```

OUTPUT:



	MERCHID	MERCHANDISE	SELLINGPRICE	SHOPID
1	8001	Theme Park T-Shirt	20	7001
2	8002	Stuffed Dragon	15	7002
3	8003	Space Keychain	5	7003
4	8004	Cowboy Hat	25	7004
5	8005	Beach Towel	18	7005
5	8006	Mug	12	7001

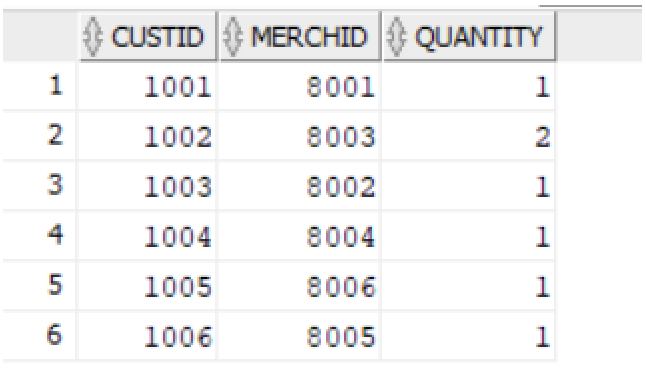
15)BUYSMERCHANDISE:

```
CREATE TABLE BuysMerchandise (
   CustID INT,
   MerchID INT,
   Quantity INT,
   PRIMARY KEY (CustID, MerchID),
   CONSTRAINT fk_buysmerch_customer FOREIGN KEY (CustID) REFERENCES
Customer(CustID),
   CONSTRAINT fk_buysmerch_merch FOREIGN KEY (MerchID) REFERENCES
Merchandise(MerchID)
);
```

INSERTING VALUES:

```
INSERT INTO BuysMerchandise VALUES (1001, 8001, 1);
INSERT INTO BuysMerchandise VALUES (1002, 8003, 2);
INSERT INTO BuysMerchandise VALUES (1003, 8002, 1);
INSERT INTO BuysMerchandise VALUES (1004, 8004, 1);
INSERT INTO BuysMerchandise VALUES (1005, 8006, 1);
INSERT INTO BuysMerchandise VALUES (1006, 8005, 1);
```

OUPUT:



	CUSTID	MERCHID	QUANTITY
1	1001	8001	1
2	1002	8003	2
3	1003	8002	1
4	1004	8004	1
5	1005	8006	1
6	1006	8005	1

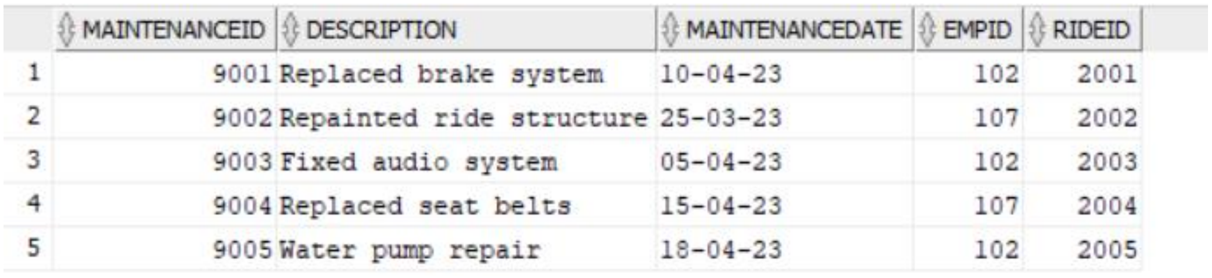
16)MAINTENANCE:

```
CREATE TABLE Maintenance (
    MaintenanceID INT PRIMARY KEY,
    Description VARCHAR(200),
    MaintenanceDate DATE,
    EmpID INT,
    RideID INT,
    CONSTRAINT fk_maintenance_employee FOREIGN KEY (EmpID) REFERENCES Employee(EmpID),
    CONSTRAINT fk_maintenance_ride FOREIGN KEY (RideID) REFERENCES Ride(RideID)
);
```

INSERTING VALUES:

```
INSERT INTO Maintenance VALUES (9001, 'Replaced brake system', TO_DATE('2023-04-10', 'YYYY-MM-DD'), 102, 2001);
INSERT INTO Maintenance VALUES (9002, 'Repainted ride structure', TO_DATE('2023-03-25', 'YYYY-MM-DD'), 107, 2002);
INSERT INTO Maintenance VALUES (9003, 'Fixed audio system', TO_DATE('2023-04-05', 'YYYY-MM-DD'), 102, 2003);
INSERT INTO Maintenance VALUES (9004, 'Replaced seat belts', TO_DATE('2023-04-15', 'YYYY-MM-DD'), 107, 2004);
INSERT INTO Maintenance VALUES (9005, 'Water pump repair', TO_DATE('2023-04-18', 'YYYY-MM-DD'), 102, 2005);
```

OUTPUT:



	MAINTENANCEID	DESCRIPTION	MAINTENANCEDATE	EMPID	RIDEID
1	9001	Replaced brake system	10-04-23	102	2001
2	9002	Repainted ride structure	25-03-23	107	2002
3	9003	Fixed audio system	05-04-23	102	2003
4	9004	Replaced seat belts	15-04-23	107	2004
5	9005	Water pump repair	18-04-23	102	2005