

1. Application Description

1.1 Introduction

The product is a database designed for usage by an Automobile Resale Dealership. A customer can either sell a pre-owned car to the dealership or buy a refurbished car from the dealership. The car sold to the dealership is outsourced for maintenance and restoration to affiliated mechanics who refurbish the car and return it to the dealership for resale. The restored vehicles are then released for sale to any customer.

1.2 Arguments

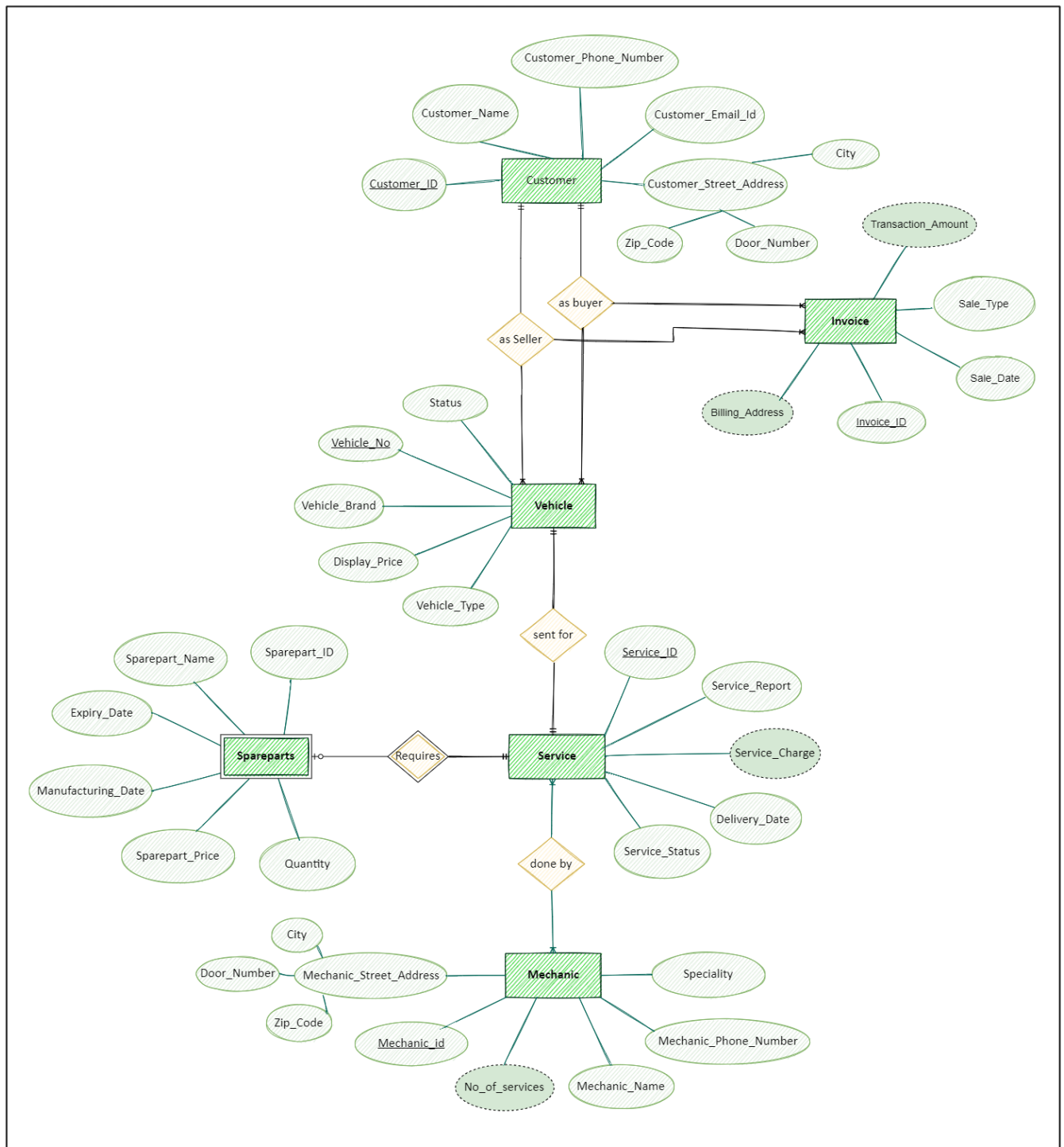
- The Customer entity has a many-to-many buying or selling relationship with the Vehicle entity, where both have total participation between them. The customer entity holds a customer's personal information.
- An Invoice is generated whenever a transaction is done and maintains a record connecting the customer to that vehicle and the transaction type (selling or buying).
- The Vehicle entity holds the information regarding the model, brand, type, purchase price, and display price of the vehicle. The selling price is derived from the purchase price and the service charge in the Service entity.
- A Vehicle sold by the customer is sent for Service. Once the service is complete, it is made available for sale to other potential customers.
- The status attribute in the Vehicle entity is set to 'for-Sale' or 'getting serviced' to denote the availability of the vehicle for purchase.
- The Service entity has relationships with the Mechanic and Spareparts entity and holds all the information related to the refurbishment
- The service is done by the mechanic and the service might require some spare parts which are accounted for in the Mechanic and Sparepart entity.
- The Mechanic entity has a many-to-many relationship with the Service entity as a mechanic might work on multiple service orders and similarly a service might require the help of multiple mechanics based on their specializations.
- The Sparepart entity is a weak entity, since not all services need a sparepart.
- The Spareparts entity has information on the parts used in the service of the vehicle. It holds data regarding the spare part name, expiry date, manufacturing date, spare part price, quantity, and sparepart_id.

1.3 Conceptual model

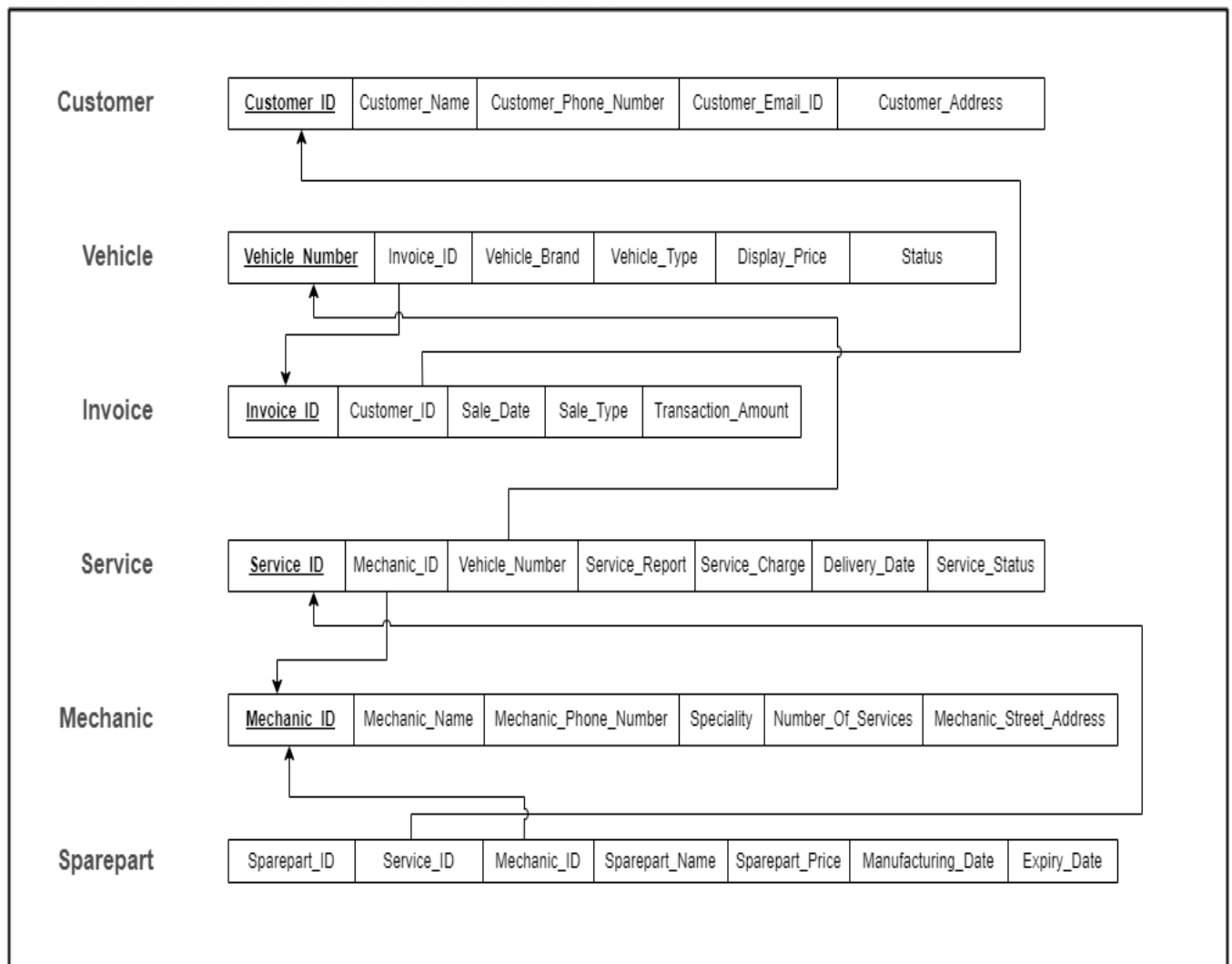
A conceptual model is a representation of a system. It consists of concepts used to help people know, understand, or simulate a subject the model represents.

1.3.1 Entity-Relationship Diagram

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects, or concepts relate to each other within a system.



2. Initial Database Schema



2.1 Entities Used

2.1.1 Customer Entity:

This entity describes Information about the customer. It tracks all the data and maintains major records like name, email, address, etc.

CUSTOMER (CUSTOMER_ID, CUSTOMER_NAME, CUSTOMER_EMAIL_ID, CUSTOMER_PHONE_NUMBER, CUSTOMER_ADDRESS, ZIP_CODE)

2.1.2 Vehicle Entity:

This entity contains basic information about the vehicles enter the dealership. Attributes like display price (if the car is ready to sell), vehicle number, etc.

VEHICLE (VEHICLE_NO, VEHICLE_BRAND, VEHICLE_TYPE, DISPLAY_PRICE, STATUS)

2.1.3 Invoice Entity:

This entity gathers data about all the billing information that had been done regarding any transactions either for spare part purchases or services or anything that is sold or bought. For example, billing address, sale date, etc.

INVOICE (INVOICE_ID, CUSTOMER_ID, SALE_DATE, SALE_TYPE, AMOUNT)

2.1.4 Service Entity:

This entity contains a database of service that has been done for the vehicle like service reports, delivery dates, service status, etc.

SERVICE (SERVICE_ID, MECHANIC_ID, SERVICE_REPORT, SERVICE_CHARGE, DELIVERY_DATE, SERVICE_STATUS, VEHICLE_NO)

2.1.5 Mechanic Entity:

This entity contains information about the mechanic and the address of the garage.

MECHANIC (MECHANIC_ID, MECHANIC_NAME, MECHANIC_PHONE_NUMBER, SPECIALTY, NO_OF_SERVICES, MECHANIC_ADDRESS)

2.1.6 Spare part Entity:

This entity contains basic information about the type of spare parts that are used for the vehicles and manufacturing date, expiry date, etc.

SPAREPART (SPAREPART_ID, SPAREPART_NAME, MANUFACTURING_DATE, EXPIRY_DATE, SPAREPART_PRICE)

3. Normalization

3.1 First Normalization Form

The first normalization form was achieved by just separating the address into separate attributes such as street address, door number, zipcode, city and state.

Customer – {**Customer ID**, Customer_Name, Customer_Street_Address, Door_Number, Zip_Code, Customer_Phone_Number, Customer_Email_Id}

Vehicle - {**Vehicle Number**, Invoice_Id, Vehicle_Brand, Vehicle_Type, Display_Price, Status}

Invoice - {**Invoice Id**, Customer_Id, Sale_Date, Sale_Type, Transaction_Amount}

Service - {**Service Id**, Mechanic_Id, Vehicle_Number, Service_Report, Service_Charge, Delivery_Date, Service_Status}

Mechanic - {**Mechanic Id**, Mechanic_Name, Mechanic_Phone_Number, Specialty, Number_Of_Services, Mechanic_Street_Address, Door_Number, Zip_Code}

Spareparts – {**Service Id**, Mechanic_Id, Sparepart_Id, Sparepart_Name, Sparepart_Price, Manufacturing_Date, Expiry_Date}

3.2 Second Normalization Form

The second normalization Form was achieved by separating attributes such as city, state and country into separate table with zipcode as the primary key to remove the partial dependencies.

Customer - {**Customer ID**, Customer_Name, Customer_Phone_Number, Customer_Email_Id, Customer_Street_Address, Door_Number, Zip_Code}

Zip_Code_Table – {**Zip Code**, City, State, Country} (We added a zip code table to avoid redundancy with the addresses)

Service - {**Service Id, Mechanic Id**, Service_Report, Service_Charge, Delivery_Date, Service_Status}

Service_Vehicle_Join - {**Service Id, Mechanic Id**, Vehicle_Number, Invoice_Id}

3.3 Third Normal Form

The third normalization form is achieved by separating transitive dependencies into separate tables. Customer phone number, email address in the Customer entity and mechanic phone number in the Mechanic entity are some of the changes made to realize this.

Customer - {**Customer ID**, Customer_Name, Customer_Street_Address, Door_Number, Zip_Code}

Customer_Phone – {**Customer ID**, Customer_Phone_Number}

Customer_Email – {Customer_ID, Customer_Email_Id}

Mechanic – {Mechanic_Id, Mechanic_Name, Specialty, Number_Of_Services}

Mechanic_Phone – {Mechanic_Id, Mechanic_Phone_Number}

Mechanic_Address – {Mechanic_Id, Mechanic_Street_Address, Door_Number, Zip_Code}

Spareparts – {Service_Id, Mechanic_Id, Sparepart_Id, Manufacturing_Date, Expiry_Date}

Sparepart_Cost – {Sparepart_Name, Sparepart_Price}

3.4 Boyce-Codd Normal Form

A relation is in Boyce/Codd Normal Form (BCNF) if whenever a non-trivial functional dependency $X \rightarrow A$ exists, then X is a Superkey.

The 3NF form also satisfies the Boyce-Codd Normal Form so no separate changes were made.

Customer - {Customer_ID, Customer_Name, Customer_Street_Address, Door_Number, Zip_Code}

Customer_Phone – {Customer_ID, Customer_Phone_Number}

Customer_Email – {Customer_ID, Customer_Email_Id}

Zip_Code_Table – {Zip_Code, City, State, Country} (We added a zip code table to avoid redundancy with the addresses)

Vehicle - {Vehicle_Number, Invoice_Id, Vehicle_Brand, Vehicle_Type, Display_Price, Status}

Invoice - {Invoice_Id, Customer_Id, Sale_Date, Sale_Type, Transaction_Amount}

Service - {Service_Id, Mechanic_Id, Service_Report, Service_Charge, Delivery_Date, Service_Status}

Service_Vehicle_Join - {Service_Id, Mechanic_Id, Vehicle_Number, Invoice_Id}

Mechanic – {Mechanic_Id, Mechanic_Name, Specialty, Number_Of_Services}

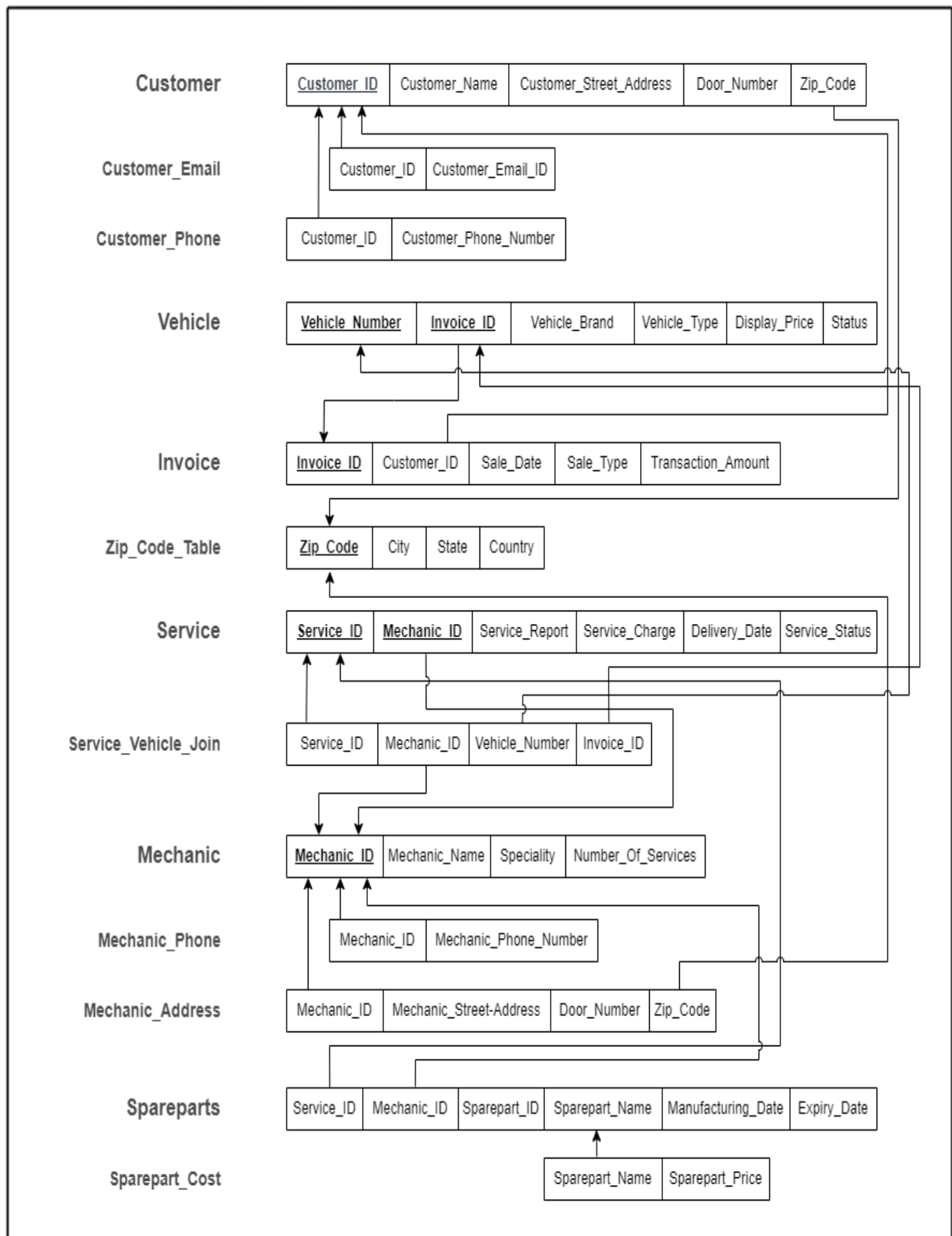
Mechanic_Phone – {Mechanic_Id, Mechanic_Phone_Number}

Mechanic_Address – {Mechanic_Id, Mechanic_Street_Address, Door_Number, Zip_Code}

Spareparts – {Service_Id, Mechanic_Id, Sparepart_Id, Sparepart_Name, Manufacturing_Date, Expiry_Date}

Sparepart_Cost – {Sparepart_Name, Sparepart_Price}

4. Final Database Schema



4.1 Entities Used

4.1.1 Customer Entity:

Customer: (Primary Key: Customer_ID, Foreign Key: Zip_Code)

Customer_Email: (Foreign Key: Customer_ID)

Customer_Phone: (Foreign Key: Customer_ID)

4.1.2 Vehicle Entity:

Vehicle: (Primary Key: Vehicle_Number)

Invoice_Vehicle_Join: (Foreign Key: Vehicle_Number, Invoice_ID)

4.1.3 Invoice Entity:

Invoice: (Primary Key: Invoice_ID, Foreign Key: Customer_ID)

4.1.4 Service Entity:

Service: (Primary Key: Service_ID, Foreign Key: Mechanic_ID)

Service_Vehicle_Join: (Foreign Key: Service_ID, Vehicle_Number)

4.1.5 Mechanic Entity:

Mechanic: (Primary Key: Mechanic_ID)

Mechanic_Phone: (Foreign Key: Mechanic_ID)

Machanic_Address: (Foreign Key: Mechanic_ID, Zip_Code)

4.1.6 Sparepart Entity:

Sparepart: (Primary Key: Sparepart_ID)

Sparepart_Cost: (Foreign Key: Sparepart_Name)

Sparepart_Service_Join: (Foreign Key: Sparepart_ID, Service_ID, Mechanic_ID)

4.1.7 Zip_Code Entity:

Zip_Code_Table: (Primary Key: Zip_Code)

4.2 Properties of a good database:

4.2.1 Data Integrity:

Data integrity is the creation and maintenance of accurate and complete data consistently throughout the software life cycle. Primary key constraints and foreign key constraints are applied to all tables to ensure this.

4.2.2 Data Redundancy:

Constraints are enforced such that there are no duplicate records in any of the tables in the database. Relationship join tables are also created to minimize data redundancy to connect various tables.

4.2.3 Data Independence:

Data independence is the ability to modify the scheme without affecting the programs and the application to be rewritten. The data in the tables are well isolated from all external applications that use it for computation or presentation.











4.2.4 Data Security:

This database application can only be accessed by users who have a profile created by the admin. The admin can provide access and specific privileges to individuals or groups of users through SQL queries.











5. Database Instance:

5.1 Database Tables:











CUSTOMER:

EDIT	CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_STREET_ADDRESS	CUSTOMER_DOOR_NUMBER	ZIP_CODE
	101744371	Kevin Richard	1710 Frericks Way	2D	45409
	101745449	Tanishq Sadanala	1710 Frericks Way	2D	45409
	101744609	Sudarshan Sowndiah	1710 Frericks Way	2B	45409
	101746193	Aswath Narayan Murthy Thenetti Bhanumurthy	1710 Frericks Way	2A	45409
	101745272	Saivarshitha Thammera	1710 Frericks Way	2C	45409
	101733455	Simon green	945 wilmington avenue	3B	45409
	101721334	James morris	firwood st 1	C	43060
	101766577	Ramesh shukla	344 patterson	8-35	43062
	101722764	Amit pandey	110 Cannonbury CT	apt F	45407
	101771221	Balaram Nayak	45329 Centerville	4A-1	45408

CUSTOMER_PHONE:

EDIT	CUSTOMER_ID	CUSTOMER_PHONE_NUMBER
	101744371	987654321
	101744609	987654331
	101745449	987654341
	101745272	987654342
	101746193	987654343
	101766577	937465874
	101733455	937428574
	101721334	9962573677
	101722764	8765354657
	101771221	9765355645












CUSTOMER_EMAIL:

EDIT	CUSTOMER_ID	CUSTOMER_EMAIL_ID
	101744371	kevin@gmail.com
	101744609	sudharshan@gmail.com
	101745449	tanishq@gmail.com
	101745272	varshitha@gmail.com
	101746193	ashwath@gmail.com
	101766577	ramesh@gmail.com
	101733455	simon@gmail.com
	101721334	james@gmail.com
	101722764	amit@gmail.com
	101771221	balram@gmail.com












VEHICLE:

EDIT	VEHICLE_NO	VEHICLE_BRAND	VEHICLE_TYPE	DISPLAY_PRICE	STATUS	INVOICE_ID
	2567635	Nissan	Sedan	-	getting serviced	567876
	2567634	bugatti	Sedan	10000	for-Sale	567657
	2567623	Benz	SUV	8000	for-Sale	587657
	2453564	Acura	Sedan	-	sold	567890
	2664566	Audi	Hatchback	-	sold	567654
	2244353	Cadillac	SUV	5000	for-Sale	576676
	2987891	Ford	Sedan	-	sold	556765
	2332423	Ford	Sedan	-	getting serviced	566765
	3456643	Dodge	Off Road	-	getting serviced	566545
	2556656	Volkswagen	Sedan	-	getting serviced	564564
	2987891	Ford	Sedan	-	sold	544765

MECHANIC:

EDIT	MECHANIC_ID	MECHANIC_NAME	SPECIALITY	NO_OF_SERVICE
	123456	Rahul Ramaraju	ac	3
	654321	Arjun Penmatsa	doors	2
	154321	Pavan vuppala	engine	1
	654323	Rick harry	motors	2
	112343	Steven james	dents	5
	122342	Peter robertson	enamel work	4
	114553	Richard Kenny	wiring	6
	114552	Bhanu Pacha	wheels	1
	122554	Aron kevinsky	vacuuming	3
	122524	Stuart Cooper	chasie	1
	85687	114553	Fixed music system	300











INVOICE:

EDIT	INVOICE_ID	CUSTOMER_ID	SALE_DATE	AMOUNT	SALE_TYPE
	567876	101744371	04/03/2022	3000	Buying
	567657	101745449	05/03/2022	4000	Buying
	587657	101744609	06/03/2022	6000	Buying
	567890	101746193	05/02/2022	3000	Selling
	567654	101745272	10/02/2019	200	Selling
	576676	101733455	06/19/2020	6000	Buying
	556765	101721334	07/23/2021	3244	Selling
	566765	101766577	09/07/2022	100	Buying
	566545	101722764	03/03/2021	400	Buying
	564564	101771221	05/03/2021	655	Buying
	544765	101744371	07/23/2020	3000	Buying

SPAREPART:

EDIT	SPAREPART_ID	SPAREPART_NAME	EXPIRY_DATE	MANUFACTURING_DATE	SERVICE_ID	MECHANIC_ID
	35440101	Headlight	07/02/2024	06/08/2019	87687	123456
	35440215	Sidemirror	04/29/2025	09/17/2020	87647	154321
	35440445	dent removal pen	02/01/2024	08/07/2022	87005	112343
	35440234	Matte black enamel	06/07/2024	09/11/2020	86789	122342
	3544222	2mm wires	11/21/2024	04/22/2019	886675	114553
	3554323	Chasie	05/06/2028	01/02/2022	790898	122524
	3554354	Front Bumper	11/13/2030	07/22/2020	790898	122524
	3554545	Engine crankshaft	05/06/2033	04/03/2022	87647	154321
	3444352	Enamel Paint	06/07/2026	02/03/2020	86789	122342
	34462543	Door Wireframes	12/22/2025	02/03/2020	790898	122524
	3556542	AC vent	09/09/2031	02/03/2022	899787	122554
	3443434	AC vent	08/09/2030	11/07/2020	87945	654323
	2433543	Engine Chain	07/03/2028	11/25/2020	87945	654323

ZIP_CODE_TABLE:

EDIT	ZIP_CODE	CITY	STATE	COUNTRY
	45409	-	Illinois	america
	45408	Dayton	North carolina	america
	45407	Dayton	New jersey	america
	45406	Dayton	Texas	america
	43058	Newark	Ohio	america
	43060	North Lewisburg	Ohio	America
	43061	Ostrander	Ohio	America
	43062	Pataskala	Illinois	America
	43011	Centerburg	ohio	America
	43029	Irwin	Wisconsin	America

5.2 Table Create Queries:

5.2.1 Customer Table

```
CREATE TABLE customer
(
    customer_id numeric(7) not null,
    customer_name varchar2(150) not null,
    customer_street_address varchar2(150) not null,
    customer_door_number varchar2(20) not null,
    zip_code numeric(6) not null,
    CONSTRAINT customer_pk primary key (customer_id)
);
alter TABLE customer add CONSTRAINT zip_code_fk foreign key (zip_code) references zip_code_table(zip_code);
```

5.2.2 Mechanic Table

```
CREATE TABLE MECHANIC
(
    mechanic_id numeric(6) not null,
    mechanic_name varchar2(150) not null,
    speciality varchar2(150),
    no_of_service int,
    CONSTRAINT mechanic_pk primary key (mechanic_id)
);
```

5.2.3 Service Table

```
CREATE TABLE service
(
    service_id numeric(10) not null,
    mechanic_id numeric(6) not null,
    service_report varchar(255) null,
    service_charge numeric(30) null,
    delivery_date DATE null,
    service_status varchar(255) not null,
    constraint service_status_check check (service_status = 'in-Progress' OR service_status = 'Completed'),
    CONSTRAINT service_pk primary key (service_id,mechanic_id),
    CONSTRAINT service_fk foreign key (mechanic_id) references mechanic(mechanic_id)
);
```

5.2.4 Mechanic_phone Table

```
CREATE TABLE mechanic_phone
(
    mechanic_id numeric(6) not null,
    mechanic_phone_number numeric(10) not null,
    CONSTRAINT mechanic_phone_fk foreign key (mechanic_id) references mechanic(mechanic_id)
);
```

5.2.5 Service_vehicle_join Table

```
CREATE TABLE service_vehicle_join (  
  service_id numeric(10) not null,  
  mechanic_id numeric(6) not null,  
  vehicle_no varchar(10) not null,  
  CONSTRAINT service_join_vehicle_fk foreign key (service_id,mechanic_id) references service(service_id,mechanic_id),  
  CONSTRAINT vehicle_join_service_fk foreign key (vehicle_no) references vehicle(vehicle_no)  
);
```

5.2.6 Sparepart Table

```
CREATE TABLE sparepart (  
  sparepart_id numeric(8) not null,  
  sparepart_name varchar2(100) not null,  
  expiry_date date not null,  
  manufacturing_date date not null,  
  CONSTRAINT sparepart_pk primary key (sparepart_id),  
  constraint sparepart_name_fk foreign key (sparepart_name) references sparepart_cost(sparepart_name)  
);
```

5.2.7 Service_sparepart_join

```
CREATE TABLE service_sparepart_join (  
  sparepart_id numeric(8) not null,  
  service_id numeric(10) not null,  
  mechanic_id numeric(6) not null,  
  constraint sparepart_id_fk foreign key (sparepart_id) references sparepart(sparepart_id),  
  CONSTRAINT service_join_sparepart_fk foreign key (service_id,mechanic_id) references service(service_id,mechanic_id)  
);
```

5.2.8 Invoice Table

```
CREATE TABLE invoice (  
  invoice_id numeric(10) not null,  
  customer_id numeric(9) not null,  
  sale_date DATE not null,  
  amount numeric(38) not null,  
  sale_type varchar(255) not null,  
  constraint sale_type_check check (sale_type = 'Selling' OR sale_type = 'Buying'),  
  CONSTRAINT invoice_pk primary key (invoice_id),  
  constraint invoice_fk foreign key (customer_id) references customer(customer_id)  
);
```

5.2.9 Zip_code_table

```
CREATE TABLE zip_code_table
(
  zip_code numeric(6) not null,
  city varchar(255),
  state varchar(255) not null,
  country varchar(255) not null,
  CONSTRAINT zip_code_table_pk primary key (zip_code)
);
```

5.2.10 Vehicle table

```
CREATE TABLE vehicle
(
  vehicle_no varchar2(10) not null,
  vehicle_brand varchar2(150) not null,
  vehicle_type varchar2(150),
  display_price numeric(30) ,
  status varchar(255) not null,
  constraint car_sale_status_check check (status = 'for-Sale' OR status = 'getting serviced'),
  CONSTRAINT vehicle_pk primary key (vehicle_no)
);
```

6. Data Manipulation:

Data manipulation is the process of changing or altering data in order to make it more readable and organized

6.1 Customer_invoice_join table using INNER JOIN:

```
select
    INVOICE_ID,
    invoice.CUSTOMER_ID as CUSTOMER_ID,
    CUSTOMER_NAME,
    AMOUNT,
    SALE_TYPE,
    concat(concat(concat(CONCAT(CUSTOMER_STREET_ADDRESS,' ' ), CUSTOMER_DOOR_NUMBER),' '), zip_code) AS Billing_Address
from
    invoice
inner join customer on invoice.customer_id = customer.customer_id
```

Results Explain Describe Saved SQL History

INVOICE_ID	CUSTOMER_ID	CUSTOMER_NAME	AMOUNT	SALE_TYPE	BILLING_ADDRESS
544765	101744371	Kevin Richard	3000	Buying	1710 Frericks Way, 2D, 45409
567876	101744371	Kevin Richard	3000	Buying	1710 Frericks Way, 2D, 45409
567657	101745449	Tanishq Sadanala	4000	Buying	1710 Frericks Way, 2D, 45409
587657	101744609	Sudarshan Sowndiah	6000	Buying	1710 Frericks Way, 2B, 45409
567890	101746193	Aswath Narayan Murthy Thenetti Bhanumurthy	3000	Selling	1710 Frericks Way, 2A, 45409
567654	101745272	Saivarshitha Thammara	200	Selling	1710 Frericks Way, 2C, 45409
576676	101733455	Simon green	6000	Buying	945 wilmington avenue, 3B, 45409
556765	101721334	James morris	3244	Selling	firwood st 1, C, 43060
566765	101766577	Ramesh shukla	100	Buying	344 patterson, 8-35, 43062
566545	101722764	Amit pandey	400	Buying	110 Cannonbury CT, apt F, 45407

6.2 Customer_Details table using LEFT JOIN:

```
select
    CUSTOMER_ID,
    CUSTOMER_NAME,
    CUSTOMER_STREET_ADDRESS ,
    CUSTOMER_DOOR_NUMBER,
    ZIP_CODE,
    CUSTOMER_EMAIL_ID,
    CUSTOMER_PHONE_NUMBER
from
    CUSTOMER LEFT JOIN
    customer_EMAIL on CUSTOMER_EMAIL.customer_id = customer.customer_id RIGHT JOIN CUSTOMER_PHONE ON CUSTOMER_PHONE.customer_id = customer.customer_id WHERE CUSTOMER_ID=101744371
```

Results Explain Describe Saved SQL History

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_STREET_ADDRESS	CUSTOMER_DOOR_NUMBER	ZIP_CODE	CUSTOMER_EMAIL_ID	CUSTOMER_PHONE_NUMBER
101744371	Kevin Richard	1710 Frericks Way	2D	45409	kevin@gmail.com	987654321

6.3 Customer_Vehicle details table using JOINS:

```
select
customer.CUSTOMER_ID as customer_id,
CUSTOMER_NAME,
SALE_TYPE,
VEHICLE_NO,
VEHICLE_BRAND,
VEHICLE_TYPE,
AMOUNT,
SALE_DATE,
invoice.INVOICE_ID as invoice_id
from
CUSTOMER LEFT JOIN
invoice on CUSTOMER.customer_id = invoice.customer_id right join vehicle on invoice.invoice_id=vehicle.invoice_id where customer.customer_id=101745272
```

Results Explain Describe Saved SQL History

CUSTOMER_ID	CUSTOMER_NAME	SALE_TYPE	VEHICLE_NO	VEHICLE_BRAND	VEHICLE_TYPE	AMOUNT	SALE_DATE	INVOICE_ID
101745272	Salvarshitha Thammera	Selling	2664566	Audi	Hatchback	200	10/02/2019	567654

6.4 Service_Report Details table using LEFT JOIN:

```
select
s2.SERVICE_ID,s2.MECHANIC_ID ,MECHANIC_NAME,SPAREPART_ID,sparepart.SPAREPART_NAME,
sparepart_cost.sparepart_price,s2.SERVICE_REPORT ,s2.SERVICE_CHARGE ,s2.SERVICE_STATUS
from SPAREPART LEFT JOIN
    sparepart_cost on sparepart.SPAREPART_NAME=sparepart_cost.SPAREPART_NAME
    join SERVICE S1 on S1.SERVICE_ID = SPAREPART.SERVICE_id
    join SERVICE S2 on S2.MECHANIC_ID=SPAREPART.MECHANIC_ID JOIN MECHANIC
    on mechanic.mechanic_id=s2.mechanic_id
WHERE S2.SERVICE_ID=87945
```

Results Explain Describe Saved SQL History

SERVICE_ID	MECHANIC_ID	MECHANIC_NAME	SPAREPART_ID	SPAREPART_NAME	SPAREPART_PRICE	SERVICE_REPORT	SERVICE_CHARGE	SERVICE_STATUS
87945	654323	Rick harry	3443434	AC vent	89	MOTOR MALFUNCTION	-	in-Progress
87945	654323	Rick harry	2433543	Engine Chain	20	MOTOR MALFUNCTION	-	in-Progress

2 rows returned in 0.01 seconds [Download](#)

6.5 Mechanic_Work Details table using FULL JOIN & RIGHT JOIN:

```
select
svj.MECHANIC_ID,MECHANIC_NAME,MECHANIC_PHONE_NUMBER,SERVICE_ID,INVOICE_ID,VEHICLE_NO,SPECIALITY
from
    service_vehicle join svj FULL JOIN mechanic m
    on svj.mechanic_id=m.mechanic_id
    RIGHT JOIN mechanic phone mp
    on m.mechanic_id=mp.mechanic_id
    where mechanic_id=114553
```

Results Explain Describe Saved SQL History

MECHANIC_ID	MECHANIC_NAME	MECHANIC_PHONE_NUMBER	SERVICE_ID	INVOICE_ID	VEHICLE_NO	SPECIALITY
114553	Richard Kenny	9379088112	886675	564564	2556656	wiring
114553	Richard Kenny	9379088112	85687	544765	2987891	wiring

6.6 Vehicle buy sell amount difference table using SUM:

```
select SUM( case when SALE_TYPE='Buying' then amount else 0 end) as Purchase_Amount, SUM( case when SALE_TYPE='Selling' then amount else 0 end) as Sales_Amount,  
SUM( case when SALE_TYPE='Selling' then amount else 0 end) - SUM( case when SALE_TYPE='Buying' then amount else 0 end) as Profit FROM INVOICE
```

Results Explain Describe Saved SQL History

PURCHASE_AMOUNT	SALES_AMOUNT	PROFIT
23155	6444	-16711

6.7 Details on vehicle sale using ORDER BY DESC

```
select  
iv.INVOICE_ID,CUSTOMER_ID,SALE_DATE,SALE_TYPE,VEHICLE_NO,VEHICLE_BRAND,VEHICLE_TYPE,AMOUNT  
from  
invoice iv LEFT JOIN vehicle v  
on iv.invoice_id=v.invoice_id ORDER BY SALE_DATE DESC
```

Results Explain Describe Saved SQL History

INVOICE_ID	CUSTOMER_ID	SALE_DATE	SALE_TYPE	VEHICLE_NO	VEHICLE_BRAND	VEHICLE_TYPE	AMOUNT
566765	101766577	09/07/2022	Buying	2332423	Ford	Sedan	100
587657	101744609	06/03/2022	Buying	2567623	Benz	SUV	6000
567657	101745449	05/03/2022	Buying	2567634	bugatti	Sedan	4000
567890	101746193	05/02/2022	Selling	2453564	Acura	Sedan	3000
567876	101744371	04/03/2022	Buying	2567635	Nissan	Sedan	3000
556765	101721334	07/23/2021	Selling	2987891	Ford	Sedan	3244
564564	101771221	05/03/2021	Buying	2556656	Volkswagen	Sedan	655
566545	101722764	03/03/2021	Buying	3456643	Dodge	Off Road	400
544765	101744371	07/23/2020	Buying	2987891	Ford	Sedan	3000
576676	101733455	06/19/2020	Buying	2244353	Cadillac	SUV	6000

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.00 seconds [Download](#)

6.8 Vehicles Count table using (COUNT, DISTINCT, CASE) :

```
select COUNT(distinct vehicle no) as Vehicles,  
COUNT(case when status='for-Sale' then 1 end) as For Sale,  
COUNT(case when status='sold' then 1 end) as Sold,  
COUNT(case when status='getting serviced' then 1 end) as In Service  
FROM VEHICLE
```

Results Explain Describe Saved SQL History

VEHICLES	FOR_SALE	SOLD	IN_SERVICE
10	3	4	4

1 rows returned in 0.00 seconds [Download](#)

6.9 Bought Vehicle Service Type table using JOIN

```
select
v.VEHICLE_NO,sale_type,sale_date,customer_id,svj.SERVICE_ID,SERVICE_REPORT,svj.MECHANIC_ID,VEHICLE_BRAND,VEHICLE_TYPE
from
vehicle v JOIN SERVICE_VEHICLE_JOIN svj
on v.vehicle_no=svj.vehicle_no and v.invoice_id=svj.invoice_id
join service s
on s.service_id=svj.service_id
join invoice i
on i.invoice_id=svj.invoice_id
where v.vehicle_no=2987891
```

Results Explain Describe Saved SQL History

VEHICLE_NO	SALE_TYPE	SALE_DATE	CUSTOMER_ID	SERVICE_ID	SERVICE_REPORT	MECHANIC_ID	VEHICLE_BRAND	VEHICLE_TYPE
2987891	Buying	07/23/2020	101744371	85687	Fixed music system	114553	Ford	Sedan

6.10 Most expensive Sparepart using MAX

```
select * from SPAREPART_COST where SPAREPART_PRICE in (select max(SPAREPART_PRICE) from SPAREPART_COST)
```

Results Explain Describe Saved SQL History

SPAREPART_NAME	SPAREPART_PRICE
Chasie	1300

1 rows returned in 0.00 seconds [Download](#)

7. Observations:

- If a database becomes larger and more complex to solve, it's better to break them down into smaller sub-categories.
- Working on the schema till it is properly normalized and accepting data will make it far easier than redesigning and reworking the schema after it throws errors during the database creation process.
- Various constraints and triggers can be used to control the data entered to tables and to auto-enter values to the database.
- Normalization is the most important and time-consuming part of creating a Database.
- Before initializing a database creation procedure, it is best to learn the normalization process and the requirements for the product clearly. This makes the created database more streamline and accessible.
- Oracle doesn't have an update cascade option. The Primary keys are supposed to be immutable, never changing, and constant. It is an excessively bad practice to have to update them.
- Oracle SQL has various functions like count, order by and joins to assist the user to manipulate data.