2023

Advance Database

Group members

Muhammad Ammar Shahid (23103156)
Adil Malik (22226228)
Ahsan Hafeez (22244388)

FINAL SUBMISSION

MUAHAMMAD AMMAR SHAHID

Table of Contents

1. Domain Description

"Dairy at Door" is an innovative online app catering to specialized dairy product delivery. Customers can personalize their experience by selecting convenient delivery days, alleviating the burden of repetitive shopping. The app emphasizes delivering fresh, premium-quality dairy items, easily meeting daily household requirements. With a user-friendly interface, it also offers customization options, potentially including organic choices, diverse milk varieties, and established brands. Beyond its products, the platform ensures a seamless journey through secure transactions, versatile payment methods, and responsive customer support. By merging technology and dairy delivery, "Dairy at Door" doesn't just save time; it enhances lifestyles by offering nourishing dairy essentials without compromising on quality, ultimately allowing users to savour life's more meaningful moments.

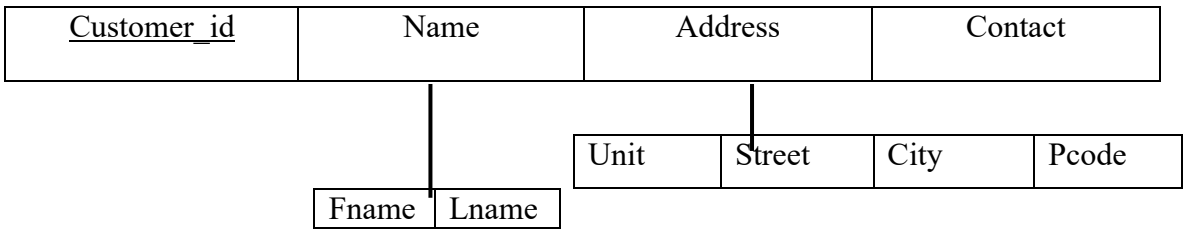
2. Database Analysis

2.1 Business Rules

- Users can sign up using the phone number and then have to select a delivery address
- Users can order instant or schedule deliveries
- A user can do product selection for a day and choose a preferred delivery time
- Subscribers will be automatically charged after each delivery
- Riders will be assigned to the delivery routes
- Riders will be notified on the delivery days
- Will have different vendors for different products
- Customers can track the riders

2.2 List of Entities and Attributes

1. Entity: Customer



Customer_id is the primary key in this table; whereas, name and address are composite attributes.

2. Entity: Employee

E_id	Name	Contact	Join_date	Service_period	Address	Type

Unit	Street	City	Pcode

E_id is the primary key in this table. Address is a composite attribute, Service_period is derived attribute from join_date and type attribute defines whether an employee is a driver or supervisor.

3. Entity: Driver

D_id	Lincense_no	V_reg_id

D_id is the primary key and V_reg_id is the foreign key from Delivery_vehicle table.

4. Entity: Supervisor

S_id	Employee_sup

S_id is the primary key

5. Entity: Delivery_vehicle

Reg_no	Make	Model

Reg_no is the primary key in this table.

6. Entity: Route

R_id	R_name	Driver_code	Supervisor_code	Radius_of_zone	Location

R_id is the primary key in the table; whereas Driver_code and Supervisor_code are the foreign keys.

7. Entity: Deliveries

Delivery_id	Delivery_time&date	Order_id	Status

Delivery_id is the peimary key and the Order_id is the foreign key.

8. Entity: Supplier

S_id	S_name	Contact	Address

S_id is the primary key.

9. Entity: Product

P_id	P_name	Category	Supplier_code

P_id is the primary key and supplier_code is the foreign key.

10. Entity: Order

O_id	Order_date	Amount	Customer_id

O_id is the primary key and Customer_id is foreign key

11. Entity: Payment

P_id	Date	Type	Order_code	Transaction_no

P_id is the primary key; whereas Type defines whether a payment is done by card or it's a subscription. Order_code is the foreign key.

12. Entity: Card

Card_no	Customer_id	Name	Valid_by

Card_no is the primary key.

13. Entity: Payment_method

Payment_id	Method

Sub_id is the primary key.

14. Entity: Reviews

Rev_id	Rating	Review_text	Date	Customer_id	Product_id

Rev_id is the primary key and customer_id, product_id are the foreign keys

2.3. Relationship, connectivity, and cardinalities

1. A User (CUSTOMER) may or may not have many ORDER but an order must be placed by the user.

[USER](0,M)	**<Places>**	(1,1) [ORDER]

A USER places a minimum 0 and maximum M ORDER.
An ORDER is placed by a minimum of 1 and a maximum of 1 USER.

2. An ORDER must charge PAYMENT

[ORDER](1,1)	**<Charge>**	(1,1) [PAYMENT]

AN ORDER charges a minimum of 1 and a maximum of 1 PAYMENT
A PAYMENT is charged by a minimum of 1 and a maximum 1 ORDER

3. A PRODUCT may or may not have an ORDER but an ORDER must have PRODUCT

[ORDER](1,M)	**<Has>**	(0,M) [PRODUCT]

A PRODUCT can have a minimum of 0 and a maximum of many ORDER
An ORDER must have at least 1 or a maximum of many PRODUCT

4. A SUPPLIER must supply atleast 1 PRODUCT

[SUPPLIER](1,M)	**<Supplies>**	(1,M) [PRODUCT]

A SUPPLIER can supply a minimum of 1 and a maximum of many PRODUCT
A PRODUCT can have 1 or more SUPPLIER

5. Every ORDER must be delivered

[ORDER](1,1)	**<Delivered by>**	(1,1) [DELIVERY]

An ORDER can be delivered a minimum and maximum of 1 by DELIVERY
A DELIVERY can have a maximum and minimum of 1 ORDER

6. A DRIVER must only drive one VECHICLE

[DRIVER](1,1)	**<Drives>**	(1,1) [VEHICLE]

A DRIVER can drive a minimum and maximum of 1 VEHICLE

A VEHICLE can be derived by a minimum and maximum of 1 DRIVER

7. A SUPERVISOR must supervise some EMPLOYEE

[SUPERVISOR](1,5)	**<Supervises>**	(1,1) [EMPLOYEE]

A SUPERVISOR can supervise a minimum of 1 and a maximum of 5 EMPLOYEES
An EMPLOYEE can be supervised by a minimum of 1 and a maximum of 1 SUPERVISOR

8. A DRIVER should do some DELIVERY

[DRIVER](0,M)	**<Do>**	(1,1) [DELIVERY]

A DRIVER can do a minimum of 0 and a maximum of many DELIVERY
A DELIVERY can be delivered by a minimum of 1 and a maximum of 1 DRIVER

9. A DRIVER must be assigned a ROUTE

[DRIVER](1,1)	**<Drive to>**	(1,1) [ROUTE]

A DRIVER can drive to a minimum of 1 and a maximum of 1 ROUTE
A ROUTE can have a minimum of 1 and a maximum of 1 DRIVER

10. A PRODUCT can have REVIEW

[PRODUCT](0,M)	**<Has>**	(1,1) [REVIEW]

A PRODUCT can have a minimum of 0 and a maximum of many REVIEWS
A REVIEW can only and must have minimum and maximum of 1 Product

11. A CUSTOMER can give REVIEW

[CUSTOMER](0,M)	**<Give>**	(1,1) [REVIEW]

A CUSTOMER can give a minimum of 0 and maximum of many REVIEWS
A REVIEW can have a minimum and maximum of 1 CUSTOMER

12. A CUSTOMER can have multiple CARDS

[CUSTOMER](0,M)	**<Has>**	(1,1) [CARDS]

A CUSTOMER can have minimum of 0 and a maximum of many CARDS
A CARD can have only 1 CUSTOMER

13. A PAYMENT must have PAYMENT_METHOD

[PAYMENT](1,1)	**<Has>**	(1,1) [PAYMENT_METHOD]

A PAYMENT can only have 1 PAYMENT_METHOD
A PAYMENT_METHOD can have only 1 PAYMENT

2.4 ERD Mapping

2.4.1 Mapping 1:1 Relationship

1. Relationship between ORDER and PAYMENT

2. Relationship between ORDER and DELIVERY

3. Relationship between DRIVER and VEHIVLE

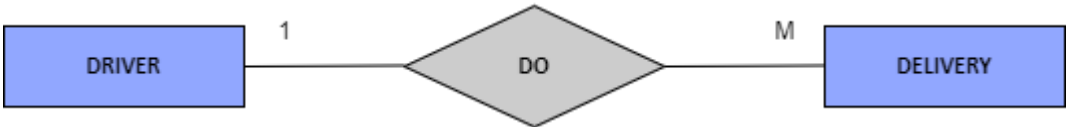
4. Relationship between DRIVER AND ROUTE

5. Relationship between PAYMENT and PAYMENT_METHOD

2.4.2 Mapping 1:M Relationship

1. Relationship between USER and ORDER

2. Relationship between DRIVER and DELIVERY



3. Relationship between SUPERVISOR and EMPLOYEE

4. Relationship between PROUCT and REVIEW

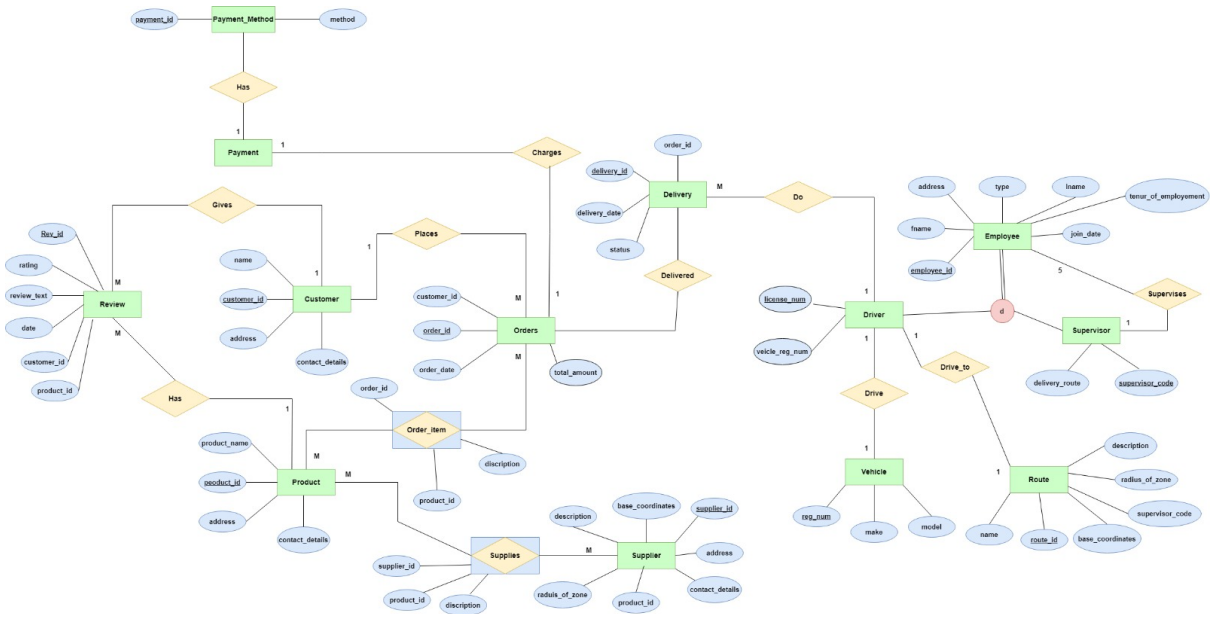
5. Relationship between CUSTOMER and REVIEW

2.4.3 Mapping M:M Relationship

1. Relationship between ORDER and PRODUCT

2. Relationship between SUPPLIER and PRODUCT

3. Database Design



4. Database Normalization:

CUSTOMER:

Customer_id	Name	Address	Contact

Functional dependencies-

Customer_id -> Name, Address, contact

Primary Key- Customer_id

The main key attribute (PK) has been identified, and all non-key attributes are fully functionally dependent on the Primary Key (contact is not multivalued because customers can have only one

phone number as it is used to verify users through OTP). There are no partial or transitive dependencies; so, the table is in 3 NF.

EMPLOYEE:

E_id	Name	Contact	Join_date	Service_period	Address	Type

Functional dependencies-

E_id -> Name, Contact, Join_date, Service_period, Address, Type

Primary Key- E_id

The table is already in 3NF as there are no partial and transitive dependencies. The main key attribute (PK) has been identified, and all non-key attributes are fully functionally dependent on the Primary key.

DRIVER:

D_id	License_no	V_reg_id	E_id

Functional dependencies-

D_id -> License_no, V_reg_id, E_id

Primary Key- E_id

Foreign Key- V_reg_id, E_id

The primary key (PK) is identified, and all non-key attributes are fully functionally dependent on the Primary key. The table is already in 3NF as there are no partial and transitive dependencies.

SUPERVISOR:

S_id	Employee_sup	E_id

Functional dependencies-

S_id -> Employee_sup, E_id

Primary Key- S_id

Foreign Key- E_id

The primary key (PK) is identified, and all non-key attributes are fully functionally dependent on the Primary key. The table is already in 3NF as there are no partial and transitive dependencies.

DELIVERY_VEHICLE:

Reg_no	Make	Model

Functional dependencies-

Reg_no -> Make, Model

Primary Key- Reg_no

The primary key (PK) is identified, and all non-key attributes are fully functionally dependent on the Primary key. The table is already in 3NF as there are no partial and transitive dependencies.

ROUTE:

R_id	R_name	Driver_code	Supervisor_code	Radius_of_zone	Location

Functional dependencies-

R_id -> R_name, Driver_code, Supervisor_code, Radius_of_zone, Location

Primary Key- R_id

Foreign Key- Driver_code, Supervisor_code

The primary key (PK) is identified, and all non-key attributes are fully functionally dependent on the Primary key. The table is already in 3NF as there are no partial and transitive dependencies.

DELIVERIES:

Delivery_id	Delivery_time&date	Order_id	Status

Functional dependencies-

Delivery_id -> Delivery_time&date, Order_id, Status

Primary Key- Delivery_id

Foreign Key- Order_id

The primary key (PK) is identified, and all non-key attributes are fully functionally dependent on the Primary key. The table is already in 3NF as there are no partial and transitive dependencies.

SUPPLIER:

S_id	S_name	Contact	Address

Functional dependencies-

S_id -> S_name, Contact, Address

Primary Key- S_id

The primary key (PK) is identified, and all non-key attributes are fully functionally dependent on the Primary key. There is one multivalued attribute Contact, To handle a multivalued attribute, a new 1:M relationship to a new entity type that contains the data for the existing attribute is used in its place.

SUPPLIER table after removing multivalued attribute

S_id	S_name	Address

SUPPLIER_CONTACT table

S_id	Contact

There aren't any transitive or partial dependencies either. So, these tables are in 3 NF.

PRODUCT:

P_id	P_name	Category	Supplier_code

Functional dependencies-

P_id -> P_name, Category, Supplier_code

Primary Key- P_id

Foreign Key- Supplier_code

Since primary key (PK) has been identified so table is in 1 NF. All the non key attributes are fully dependant on the PK except Category; so, a separate table is needed to make the existing table into 2 NF.

CATEGORY table

Category_code	Name

New PRODUCT table

P_id	P_name	Category_code	Supplier_code

Where Category_code will be a foreign key, hence all the non key attributes are fully dependent on the PK. Since there aren't any transitive or partial dependencies either. So, these tables are in 3 NF.

ORDER:

O_id	Order_date	Amount	Customer_id

Functional dependencies-

O_id -> Order_date, Amount, Customer_id

Primary Key- O_id

Foreign Key- Customer_id

The primary key (PK) is identified, and all non-key attributes are fully functionally dependent on the Primary key. The table is already in 3NF as there are no partial and transitive dependencies.

PAYMENT:

P_id	Date	Type	Order_code	Transaction_no

Functional dependencies-

P_id -> Date, Type, Order_code, Transaction_no

Primary Key- P_id

Foreign Key- Order_code

The primary key (PK) is identified, and all non-key attributes are fully functionally dependent on the Primary key. The table is already in 3NF as there are no partial and transitive dependencies.

CARD:

Card_no	Name	Valid_by	Customer_id

Functional dependencies-

Card_no-> Name, Valid_by, Customer_id

Primary Key- Card_no

Foreign Key- Customer_id

The primary key (PK) is identified, and all non-key attributes are fully functionally dependent on the Primary key. The table is already in 3NF as there are no partial and transitive dependencies.

PAYMENT_METHOD:

Payment_id	Method

Functional dependencies-

Sub_id -> Method

Primary Key- Payment_id

The primary key (PK) is identified, and all non-key attributes are fully functionally dependent on the Primary key. The table is already in 3NF as there are no partial and transitive dependencies.

REVIEWS:

Rev_id	Rating	Review_text	Date	Customer_id	Product_id

Functional dependencies-

Rev_id -> Rating, Review_text, Date, Customer_id, Product_id

Primary Key- Rev_id

Foreign Key- Customer_id, Product_id

The primary key (PK) is identified, and all non-key attributes are fully functionally dependent on the Primary key. The table is already in 3NF as there are no partial and transitive dependencies.

ASSOCIATIVE OR BRIDGE ENTITY

Order and Product entities have M:M relationship. To manage this, a Bridge or an Associative entity is introduced. This new entity is formed by combining the PK of Product and Order, which allows the effective representation of the relationship. The Bridge entity also include supplementary attributes.

Associative Entity: Order_item

Primary Key- Order_id, Product_id

Order_id	Product_id	Description

Similarly, Supplier and Product have M:M relation. To break this relation an Associative entity is introduced.

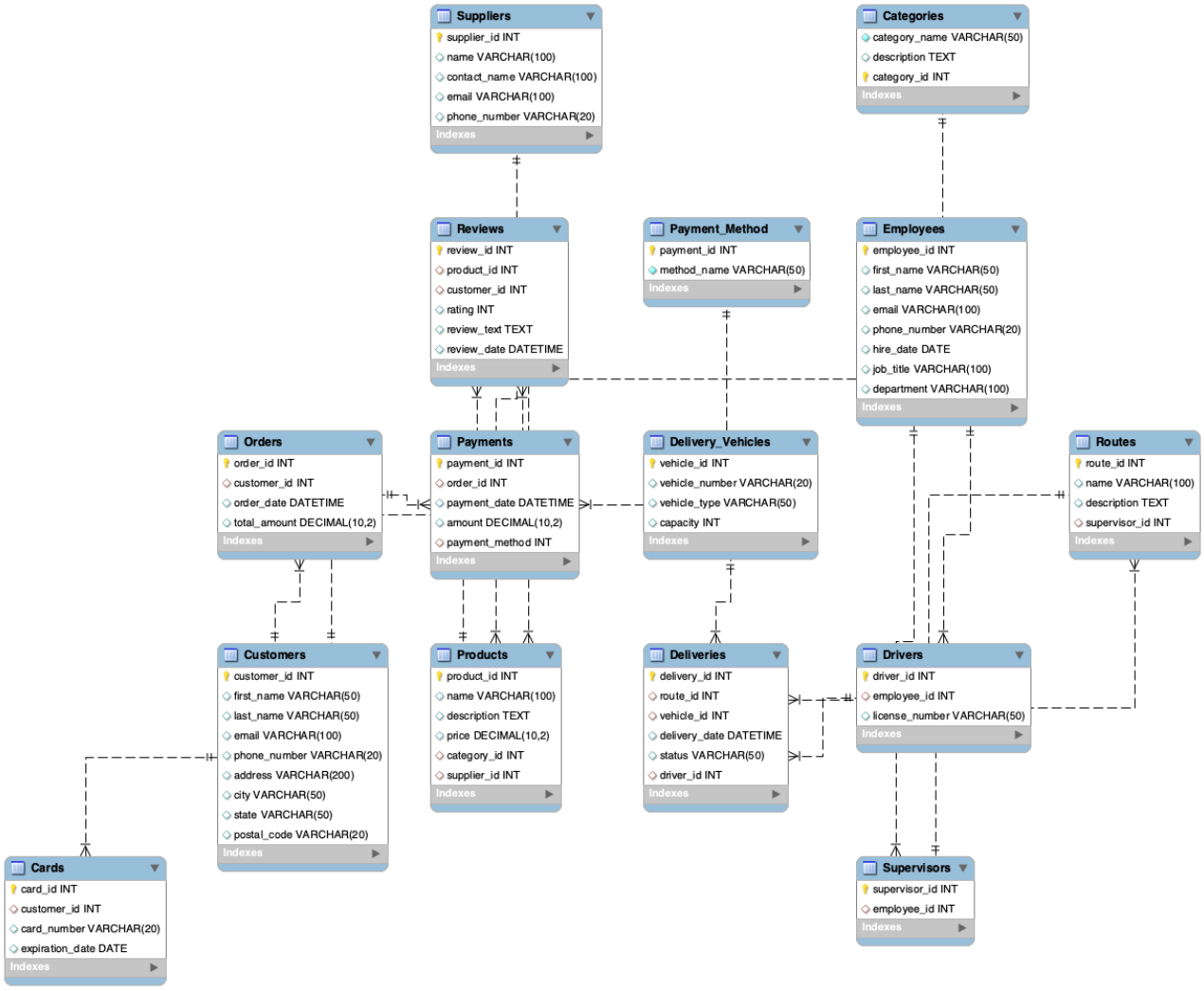
Associative Entity: Supplies

Primary Key- Product_id, Supplier_id

Supplier_id	Product_id	Description

Relational Schema

The DBMS was designed in MySQL WorkBench using MySQL dialect of SQL. Following figure show the relational schema for the datebase.



5. Database Implementation

I. Table Creation

Through this section, we'll shed some light on how the database design was implemented. Firstly, let's look at how the tables were created, one by one.

a. Customer

(Aadil)

```
CREATE TABLE Customers ( customer_id INT PRIMARY KEY, first_name
VARCHAR(50), last_name VARCHAR(50), email VARCHAR(100) UNIQUE,
phone_number VARCHAR(20), address VARCHAR(200), city VARCHAR(50), state
VARCHAR(50), postal_code VARCHAR(20) );
```

b. Employee

(Aadil)

```
CREATE TABLE Employees ( employee_id INT PRIMARY KEY, first_name
VARCHAR(50), last_name VARCHAR(50), email VARCHAR(100) UNIQUE,
phone_number VARCHAR(20), hire_date DATE, job_title VARCHAR(100),
department VARCHAR(100) );
```

c. Driver

(Ammar)

```
CREATE TABLE Drivers ( driver_id INT PRIMARY KEY, employee_id INT,
license_number VARCHAR(50) UNIQUE, CONSTRAINT fk_employee_driver
FOREIGN KEY (employee_id) REFERENCES Employees (employee_id) );
```

d. Supervisor

(Ahsan)

```
CREATE TABLE Supervisors ( supervisor_id INT PRIMARY KEY, employee_id
INT, CONSTRAINT fk_employee_supervisor FOREIGN KEY (employee_id)
REFERENCES Employees (employee_id) );
```

e. Delivery_vehicle

(Ammar)

```
CREATE TABLE Delivery_Vehicles ( vehicle_id INT PRIMARY KEY,
vehicle_number VARCHAR(20) UNIQUE, vehicle_type VARCHAR(50), capacity
INT );
```

f. Route

(Ahsan)

```
CREATE TABLE Routes ( route_id INT PRIMARY KEY, name VARCHAR(100),
description TEXT );
```

g. Deliveries
(Aadil)

```
CREATE TABLE Deliveries ( delivery_id INT PRIMARY KEY, route_id INT,
vehicle_id INT, delivery_date DATETIME, status VARCHAR(50), CONSTRAINT
fk_route_delivery FOREIGN KEY (route_id) REFERENCES Routes (route_id),
CONSTRAINT fk_vehicle_delivery FOREIGN KEY (vehicle_id) REFERENCES
Delivery_Vehicles (vehicle_id) );
```

h. Supplier
(Ammar)

```
CREATE TABLE Suppliers ( supplier_id INT PRIMARY KEY, name
VARCHAR(100), contact_name VARCHAR(100), email VARCHAR(100),
phone_number VARCHAR(20) );
```

i. Product
(Ammar)

```
CREATE TABLE Products ( product_id INT PRIMARY KEY, name VARCHAR(100),
description TEXT, price DECIMAL(10, 2), category VARCHAR(50),
supplier_id INT, CONSTRAINT fk_supplier_product FOREIGN KEY
(supplier_id) REFERENCES Suppliers (supplier_id) );
```

j. Order
(Aadil)

```
CREATE TABLE Orders ( order_id INT PRIMARY KEY, customer_id INT,
order_date DATETIME, total_amount DECIMAL(10, 2), CONSTRAINT
fk_customer_order FOREIGN KEY (customer_id) REFERENCES Customers
(customer_id) );
```

k. Payment
(Aadil)

```
CREATE TABLE Payments ( payment_id INT PRIMARY KEY, order_id INT,
payment_date DATETIME, amount DECIMAL(10, 2), payment_method
VARCHAR(50), CONSTRAINT fk_order_payment FOREIGN KEY (order_id)
REFERENCES Orders (order_id) );
```

l. Card
(Ahsan)

```
CREATE TABLE Cards ( card_id INT PRIMARY KEY, customer_id INT,
card_number VARCHAR(20), expiration_date DATE, CONSTRAINT
```

```
fk_customer_card FOREIGN KEY (customer_id) REFERENCES Customers
(customer_id) );
```

m. Reviews
(Ammar)

```
CREATE TABLE Reviews ( review_id INT PRIMARY KEY, product_id INT,
customer_id INT, rating INT, review_text TEXT, review_date DATETIME,
CONSTRAINT fk_product_review FOREIGN KEY (product_id) REFERENCES
Products (product_id), CONSTRAINT fk_customer_review FOREIGN KEY
(customer_id) REFERENCES Customers (customer_id) );
```

n. Categories
(Ammar)

```
CREATE TABLE Categories ( category_name VARCHAR(50) PRIMARY KEY,
description TEXT );
```

o. Payment_Method
(Ahsan)

```
CREATE TABLE Payment_Method ( payment_id INT PRIMARY KEY, method_name
VARCHAR(50) );
```

II. Data Insertion

a. Customers
(Aadil)

```
INSERT INTO Customers (customer_id, first_name, last_name, email,
phone_number, address, city, state, postal_code) VALUES (1, 'John',
'Doe', 'john@example.com', '123-456-7890', '123 Main St', 'Cityville',
'CA', '12345'), (2, 'Jane', 'Smith', 'jane@example.com', '234-567-
8901', '456 Elm Ave', 'Townsville', 'NY', '23456'), (3, 'Michael',
'Lee', 'michael@example.com', '345-678-9012', '789 Oak Ln',
'Villagetown', 'TX', '34567'), (4, 'Emily', 'Johnson',
'emily@example.com', '456-789-0123', '567 Maple Dr', 'Hometown', 'FL',
'45678'), (5, 'Robert', 'Brown', 'robert@example.com', '567-890-1234',
'890 Pine St', 'Hamletville', 'IL', '56789'), (6, 'Sarah', 'Williams',
'sarah@example.com', '678-901-2345', '345 Cedar Rd', 'Suburbia', 'GA',
'67890'), (7, 'David', 'Davis', 'david@example.com', '789-012-3456',
'678 Oak Rd', 'Countrytown', 'NC', '78901'), (8, 'Olivia', 'Jones',
'olivia@example.com', '890-123-4567', '456 Pine Ave', 'Outskirts',
'WA', '89012');
```

customer_id	first_name	last_name	email	phone_number	address	city	state	postal_code
1	John	Doe	john@example.com	123-456-7890	123 Main St	Cityville	CA	12345
2	Jane	Smith	jane@example.com	234-567-8901	456 Elm Ave	Townsville	NY	23456
3	Michael	Lee	michael@example.com	345-678-9012	789 Oak Ln	Villagetown	TX	34567
4	Emily	Johnson	emily@example.com	456-789-0123	567 Maple Dr	Hometown	FL	45678
5	Robert	Brown	robert@example.com	567-890-1234	890 Pine St	Hamletville	IL	56789
6	Sarah	Williams	sarah@example.com	678-901-2345	345 Cedar Rd	Suburbia	GA	67890
7	David	Davis	david@example.com	789-012-3456	678 Oak Rd	Countrytown	NC	78901
8	Olivia	Jones	olivia@example.com	890-123-4567	456 Pine Ave	Outskirts	WA	89012
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

b. Employees

(Aadil)

```
INSERT INTO Employees (employee_id, first_name, last_name, email, phone_number, hire_date, job_title, department) VALUES (1, 'Mark', 'Smith', 'mark@example.com', '123-456-7890', '2020-01-15', 'Manager', 'Operations'), (2, 'Emily', 'Davis', 'emily@example.com', '234-567-8901', '2021-03-20', 'Driver', 'Logistics'), (3, 'Andrew', 'Johnson', 'andrew@example.com', '345-678-9012', '2019-11-10', 'Supervisor', 'Supervision'), (4, 'Jessica', 'Brown', 'jessica@example.com', '456-789-0123', '2022-05-05', 'Driver', 'Logistics'), (5, 'Daniel', 'Lee', 'daniel@example.com', '567-890-1234', '2020-08-28', 'Driver', 'Logistics'), (6, 'Sophia', 'Miller', 'sophia@example.com', '678-901-2345', '2023-01-10', 'Supervisor', 'Supervision'), (7, 'William', 'Wilson', 'william@example.com', '789-012-3456', '2022-02-15', 'Driver', 'Logistics'), (8, 'Emma', 'Moore', 'emma@example.com', '890-123-4567', '2021-06-02', 'Driver', 'Logistics');
```

employee_id	first_name	last_name	email	phone_number	hire_date	job_title	department
1	Mark	Smith	mark@example.com	123-456-7890	2020-01-15	Manager	Operations
2	Emily	Davis	emily@example.com	234-567-8901	2021-03-20	Driver	Logistics
3	Andrew	Johnson	andrew@example.com	345-678-9012	2019-11-10	Supervisor	Supervision
4	Jessica	Brown	jessica@example.com	456-789-0123	2022-05-05	Driver	Logistics
5	Daniel	Lee	daniel@example.com	567-890-1234	2020-08-28	Driver	Logistics
6	Sophia	Miller	sophia@example.com	678-901-2345	2023-01-10	Supervisor	Supervision
7	William	Wilson	william@example.com	789-012-3456	2022-02-15	Driver	Logistics
8	Emma	Moore	emma@example.com	890-123-4567	2021-06-02	Driver	Logistics
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

c. Drivers

(Ammar)

```
INSERT INTO Drivers (driver_id, employee_id, license_number) VALUES (1, 2, 'DL12345'), (2, 4, 'DL23456'), (3, 5, 'DL34567'), (4, 7, 'DL45678');
```

driver_id	employee_id	license_numb...
1	2	DL12345
2	4	DL23456
3	5	DL34567
4	7	DL45678
NULL	NULL	NULL

d. Supervisors
(Ahsan)

```
INSERT INTO Supervisors (supervisor_id, employee_id) VALUES (1, 3), (2, 6);
```

supervisor_id	employee_id
1	3
2	6
NULL	NULL

e. Delivery_Vehicles
(Ammar)

```
INSERT INTO Delivery_Vehicles (vehicle_id, vehicle_number, vehicle_type, capacity) VALUES (1, 'VDL123', 'Van', 10), (2, 'TRL234', 'Truck', 20), (3, 'MCY345', 'Motorcycle', 2), (4, 'SCT456', 'Scooter', 1);
```

vehicle_id	vehicle_numb...	vehicle_type	capacity
1	VDL123	Van	10
2	TRL234	Truck	20
3	MCY345	Motorcycle	2
4	SCT456	Scooter	1
NULL	NULL	NULL	NULL

f. Routes
(Ahsan)

```
INSERT INTO Routes (route_id, name, description) VALUES (1, 'Route A',
'City center route'), (2, 'Route B', 'Suburban route'), (3, 'Route C',
'Rural route'), (4, 'Route D', 'Industrial area route');
```

route_id	name	description
1	Route A	City center route
2	Route B	Suburban route
3	Route C	Rural route
4	Route D	Industrial area route
NULL	NULL	NULL

g. Deliveries
(Ammar)

```
INSERT INTO Deliveries (delivery_id, route_id, vehicle_id,
delivery_date, status) VALUES (1, 1, 1, '2023-07-10 08:00:00',
'Delivered'), (2, 2, 2, '2023-07-11 10:00:00', 'Pending'), (3, 3, 3,
'2023-07-12 12:00:00', 'Delivered'), (4, 4, 4, '2023-07-13 14:00:00',
'Pending'), (5, 1, 1, '2023-07-14 16:00:00', 'Delivered'), (6, 2, 2,
'2023-07-15 18:00:00', 'Pending'), (7, 3, 3, '2023-07-16 20:00:00',
'Delivered'), (8, 4, 4, '2023-07-17 22:00:00', 'Pending');
```

delivery_id	route_id	vehicle_id	delivery_date	status	driver_id
1	1	1	2023-07-10 08:00:00	Delivered	3
2	2	2	2023-07-11 10:00:00	Pending	1
3	3	3	2023-07-12 12:00:00	Delivered	3
4	4	4	2023-07-13 14:00:00	Pending	4
5	1	1	2023-07-14 16:00:00	Delivered	1
6	2	2	2023-07-15 18:00:00	Pending	1
7	3	3	2023-07-16 20:00:00	Delivered	2
8	4	4	2023-07-17 22:00:00	Pending	3
NULL	NULL	NULL	NULL	NULL	NULL

h. Suppliers
(Ammar)

```
INSERT INTO Suppliers (supplier_id, name, contact_name, email,
phone_number) VALUES (1, 'Farm Fresh', 'John Farmer',
'farmfresh@example.com', '123-456-7890'), (2, 'Dairy Delights', 'Emily
Dairy', 'dairydelights@example.com', '234-567-8901'), (3, 'Organic Milk
Co.', 'Alice Organic', 'organicmilk@example.com', '345-678-9012'), (4,
'Local Dairy', 'Mike Local', 'localdairy@example.com', '456-789-0123'),
```

```
(5, 'Health Foods', 'Sarah Health', 'healthfoods@example.com', '567-
890-1234'), (6, 'Natural Products', 'David Natural',
'naturalproducts@example.com', '678-901-2345'), (7, 'Fresh Milk',
'Laura Fresh', 'freshmilk@example.com', '789-012-3456'), (8, 'Quality
Dairy', 'Robert Quality', 'qualitydairy@example.com', '890-123-4567');
```

supplier_id	name	contact_name	email	phone_number
1	Farm Fresh	John Farmer	farmfresh@example.com	123-456-7890
2	Dairy Delights	Emily Dairy	dairydelights@example.com	234-567-8901
3	Organic Milk Co.	Alice Organic	organicmilk@example.com	345-678-9012
4	Local Dairy	Mike Local	localdairy@example.com	456-789-0123
5	Health Foods	Sarah Health	healthfoods@example.com	567-890-1234
6	Natural Products	David Natural	naturalproducts@example.com	678-901-2345
7	Fresh Milk	Laura Fresh	freshmilk@example.com	789-012-3456
8	Quality Dairy	Robert Quality	qualitydairy@example.com	890-123-4567
NULL	NULL	NULL	NULL	NULL

i. Products

(Ammar)

```
INSERT INTO Products (product_id, name, description, price, category,
supplier_id) VALUES (1, 'Whole Milk', 'Fresh whole milk', 3.99,
'Dairy', 1), (2, 'Skim Milk', 'Fat-free skim milk', 2.99, 'Dairy', 2),
(3, 'Soy Milk', 'Dairy-free soy milk', 4.49, 'Plant-Based', 3), (4,
'Almond Milk', 'Nutty almond milk', 4.99, 'Plant-Based', 4), (5, 'Oat
Milk', 'Creamy oat milk', 3.79, 'Plant-Based', 5), (6, 'Cheese',
'Assorted cheese varieties', 5.99, 'Dairy', 1), (7, 'Yogurt', 'Various
yogurt flavors', 2.49, 'Dairy', 2), (8, 'Ice Cream', 'Delicious ice
cream flavors', 6.99, 'Desserts', 3);
```

product_...	name	description	price	category	supplier_id
1	Whole Milk	Fresh whole milk	3.99	Dairy	1
2	Skim Milk	Fat-free skim milk	2.99	Dairy	2
3	Soy Milk	Dairy-free soy milk	4.49	Plant-Based	3
4	Almond Milk	Nutty almond milk	4.99	Plant-Based	4
5	Oat Milk	Creamy oat milk	3.79	Plant-Based	5
6	Cheese	Assorted cheese varieties	5.99	Dairy	1
7	Yogurt	Various yogurt flavors	2.49	Dairy	2
8	Ice Cream	Delicious ice cream flavors	6.99	Desserts	3
NULL	NULL	NULL	NULL	NULL	NULL

j. Orders

(Aadil)

```
INSERT INTO Orders (order_id, customer_id, order_date, total_amount)
VALUES (1, 1, '2023-07-01 09:00:00', 10.97), (2, 2, '2023-07-02
10:00:00', 4.49), (3, 3, '2023-07-03 11:00:00', 7.98), (4, 4, '2023-07-
04 12:00:00', 12.50), (5, 5, '2023-07-05 13:00:00', 6.75), (6, 6,
```

```
'2023-07-06 14:00:00', 8.95), (7, 7, '2023-07-07 15:00:00', 15.25), (8,
8, '2023-07-08 16:00:00', 11.80);
```

order_id	customer_id	order_date	total_amou...
1	1	2023-07-01 09:00:00	10.97
2	2	2023-07-02 10:00:00	4.49
3	3	2023-07-03 11:00:00	7.98
4	4	2023-07-04 12:00:00	12.50
5	5	2023-07-05 13:00:00	6.75
6	6	2023-07-06 14:00:00	8.95
7	7	2023-07-07 15:00:00	15.25
8	8	2023-07-08 16:00:00	11.80
NULL	NULL	NULL	NULL

k. Payments

(Aadil)

```
INSERT INTO Payments (payment_id, order_id, payment_date, amount,
payment_method) VALUES (1, 1, '2023-07-02 09:30:00', 10.97, 'Credit
Card'), (2, 2, '2023-07-03 10:30:00', 4.49, 'PayPal'), (3, 3, '2023-07-
04 11:30:00', 7.98, 'Credit Card'), (4, 4, '2023-07-05 12:30:00',
12.50, 'Cash'), (5, 5, '2023-07-06 13:30:00', 6.75, 'Credit Card'), (6,
6, '2023-07-07 14:30:00', 8.95, 'PayPal'), (7, 7, '2023-07-08
15:30:00', 15.25, 'Credit Card'), (8, 8, '2023-07-09 16:30:00', 11.80,
'Cash');
```

payment_id	order_id	payment_date	amount	payment_meth...
1	1	2023-07-02 09:30:00	10.97	Credit Card
2	2	2023-07-03 10:30:00	4.49	PayPal
3	3	2023-07-04 11:30:00	7.98	Credit Card
4	4	2023-07-05 12:30:00	12.50	Cash
5	5	2023-07-06 13:30:00	6.75	Credit Card
6	6	2023-07-07 14:30:00	8.95	PayPal
7	7	2023-07-08 15:30:00	15.25	Credit Card
8	8	2023-07-09 16:30:00	11.80	Cash
NULL	NULL	NULL	NULL	NULL

l. Cards

(Ahsan)

```
INSERT INTO Cards (card_id, customer_id, card_number, expiration_date)
VALUES (1, 1, '1234-5678-9012-3456', '2025-12-31'), (2, 2, '2345-6789-
```

```
0123-4567', '2024-10-15'), (3, 3, '3456-7890-1234-5678', '2026-05-20'),
(4, 4, '4567-8901-2345-6789', '2023-08-05'), (5, 5, '5678-9012-3456-
7890', '2025-06-30'), (6, 6, '6789-0123-4567-8901', '2024-09-22'), (7,
7, '7890-1234-5678-9012', '2026-02-17'), (8, 8, '8901-2345-6789-0123',
'2023-11-10');
```

card_id	customer_id	card_number	expiration_da...
1	1	1234-5678-9012-3456	2025-12-31
2	2	2345-6789-0123-4567	2024-10-15
3	3	3456-7890-1234-5678	2026-05-20
4	4	4567-8901-2345-6789	2023-08-05
5	5	5678-9012-3456-7890	2025-06-30
6	6	6789-0123-4567-8901	2024-09-22
7	7	7890-1234-5678-9012	2026-02-17
8	8	8901-2345-6789-0123	2023-11-10
NULL	NULL	NULL	NULL

m. Reviews (Ammar)

```
INSERT INTO Reviews (review_id, product_id, customer_id, rating,
review_text, review_date) VALUES (1, 1, 1, 4, 'Great product!', '2023-
07-05 09:00:00'), (2, 2, 2, 3, 'Average product.', '2023-07-06
10:00:00'), (3, 3, 3, 5, 'Excellent quality!', '2023-07-07 11:00:00'),
(4, 4, 4, 2, 'Not satisfied.', '2023-07-08 12:00:00'), (5, 5, 5, 5,
'Love it!', '2023-07-09 13:00:00'), (6, 6, 6, 4, 'Delicious cheese!',
'2023-07-10 14:00:00'), (7, 7, 7, 3, 'Good yogurt options.', '2023-07-
11 15:00:00'), (8, 8, 8, 5, 'Best ice cream ever!', '2023-07-12
16:00:00');
```

review_id	product_...	customer_id	rating	review_text	review_date
1	1	1	4	Great product!	2023-07-05 09:00:00
2	2	2	3	Average product.	2023-07-06 10:00:00
3	3	3	5	Excellent quality!	2023-07-07 11:00:00
4	4	4	2	Not satisfied.	2023-07-08 12:00:00
5	5	5	5	Love it!	2023-07-09 13:00:00
6	6	6	4	Delicious cheese!	2023-07-10 14:00:00
7	7	7	3	Good yogurt options.	2023-07-11 15:00:00
8	8	8	5	Best ice cream ever!	2023-07-12 16:00:00
NULL	NULL	NULL	NULL	NULL	NULL

n. Categories (Ahsan)

```
INSERT INTO Categories (category_name, description) VALUES ('dairy',
'Products made from animal milk'), ('plant_based', 'Products derived
from plant sources'), ('desserts', 'Sweet treats and desserts');
```

category_name	description	category_id
dairy	Products made from animal milk	1
plant_based	Products derived from plant sources	2
desserts	Sweet treats and desserts	3
NULL	NULL	NULL

p. Payment_Method (Ahsan)

```
INSERT INTO Payment_Method (payment_id, method_name) VALUES (1, 'paypal'), (2, 'credit card');
```

payment_id	method_name
1	paypal
2	credit card
3	cash
NULL	NULL

6. Constraint Validation:

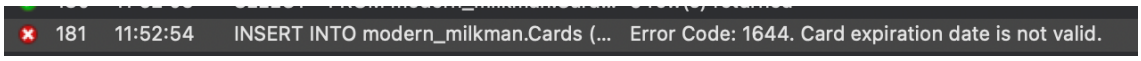
Through this section, we will place a certain number of constraints on different things and then validate them. We will be using stored procedures and delimiters in MySQL for this purpose.

a. Card Expiry Date Validation:

```
DELIMITER // CREATE PROCEDURE check_card_expiration(date_to_check DATE)
BEGIN IF date_to_check <= CURRENT_DATE() THEN SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Card expiration date is not valid.'; END IF; END;
// DELIMITER ;
```

```
DELIMITER // CREATE TRIGGER trg_check_card_expiration BEFORE INSERT ON
modern_milkman.Cards FOR EACH ROW BEGIN CALL
check_card_expiration(NEW.expiration_date); END; // DELIMITER ;
```

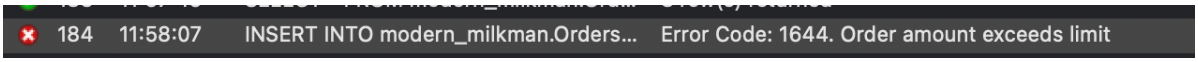
```
INSERT INTO modern_milkman.Cards (card_id, expiration_date)
VALUES (2, '2022-12-31');
```



b. Order Limit Check

```
DELIMITER // CREATE TRIGGER `trg_check_order_amount` BEFORE INSERT ON
`modern_milkman`.`Orders` FOR EACH ROW BEGIN IF NEW.total_amount > 1000
THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Order amount exceeds
limit'; END IF; END; // DELIMITER ;
```

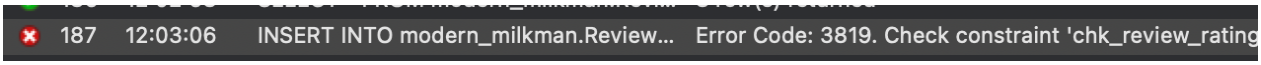
```
INSERT INTO modern_milkman.Orders (order_id, customer_id, order_date,
total_amount)
VALUES (9, 1, '2023-08-23', 1500.00);
```



c. Rating Between 1 and 5

```
ALTER TABLE `modern_milkman`.`Reviews` ADD CONSTRAINT
`chk_review_rating` CHECK (`rating` BETWEEN 1 AND 5);
```

```
INSERT INTO modern_milkman.Reviews (review_id, product_id, customer_id,
rating, review_text, review_date) VALUES (1, 1, 1, 6, 'Great product!',
'2023-08-23 12:00:00');
```



d. Unique Vehicle Number

```
ALTER TABLE `modern_milkman`.`Delivery_Vehicles` ADD CONSTRAINT
`uc_vehicle_number` UNIQUE (`vehicle_number`);
```

e. Vehicle Capacity never 0

```
ALTER TABLE `modern_milkman`.`Delivery_Vehicles` ADD CONSTRAINT
`chk_vehicle_capacity` CHECK (`capacity` > 0);
```

7. Select Queries to Retrieve Data:

a. Select Query to retrieve fname, lname, email and total amount spent on orders (Aadil):

```
SELECT c.first_name, c.last_name, c.email, SUM(o.total_amount) AS
total_spent FROM Customers c LEFT JOIN Orders o ON c.customer_id =
o.customer_id GROUP BY c.customer_id, c.first_name, c.last_name,
c.email ORDER BY total_spent DESC;
```

first_name	last_name	email	total_spent
David	Davis	david@example.com	15.25
Emily	Johnson	emily@example.com	12.50
Olivia	Jones	olivia@example.com	11.80
John	Doe	john@example.com	10.97
Sarah	Williams	sarah@example.com	8.95
Michael	Lee	michael@example.com	7.98
Robert	Brown	robert@example.com	6.75
Jane	Smith	jane@example.com	4.49

b. List Categories with Average product prices (Aadil):

```
SELECT cat.category_name, AVG(prod.price) AS average_price FROM
Categories cat LEFT JOIN Products prod ON cat.category_id =
prod.category_id GROUP BY cat.category_id, cat.category_name ORDER BY
average_price DESC;
```

category_name	average_price
desserts	6.990000
plant_based	4.423333
dairy	3.865000

c. Count Order per Customer (Ahsan)

```
SELECT c.customer_id, c.first_name, c.last_name, COUNT(o.order_id) AS
order_count
FROM Customers c
LEFT JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name;
```

customer_id	first_name	last_name	order_count
1	John	Doe	1
2	Jane	Smith	1
3	Michael	Lee	1
4	Emily	Johnson	1
5	Robert	Brown	1
6	Sarah	Williams	1
7	David	Davis	1
8	Olivia	Jones	1

d. Find Latest Deliveries (Ahsan)

```
SELECT delivery_id, route_id, vehicle_id, delivery_date, status FROM
Deliveries ORDER BY delivery_date DESC LIMIT 5;
```

delivery_id	route_id	vehicle_id	delivery_date	status
8	4	4	2023-07-17 22:00:00	Pending
7	3	3	2023-07-16 20:00:00	Delivered
6	2	2	2023-07-15 18:00:00	Pending
5	1	1	2023-07-14 16:00:00	Delivered
4	4	4	2023-07-13 14:00:00	Pending
NULL	NULL	NULL	NULL	NULL

e. Drivers with the highest delivery count (Ammar)

```
SELECT d.driver_id, e.first_name, e.last_name, COUNT(dl.delivery_id) AS
delivery_count FROM Drivers d JOIN Employees e ON d.employee_id =
e.employee_id LEFT JOIN Deliveries dl ON d.driver_id = dl.driver_id
GROUP BY d.driver_id, e.first_name, e.last_name ORDER BY delivery_count
DESC;
```

driver_id	first_name	last_name	delivery_count
1	Emily	Davis	3
3	Daniel	Lee	3
2	Jessica	Brown	1
4	William	Wilson	1

f. Top 3 Spending Customers (Ammar)

```
SELECT c.customer_id, c.first_name, c.last_name, SUM(o.total_amount) AS
total_spent FROM Customers c LEFT JOIN Orders o ON c.customer_id =
o.customer_id GROUP BY c.customer_id, c.first_name, c.last_name ORDER
BY total_spent DESC LIMIT 10;
```

customer_id	first_name	last_name	total_spent
7	David	Davis	15.25
4	Emily	Johnson	12.50
8	Olivia	Jones	11.80

g. Retrieve Supervisors and their Route Count (Ammar)

```
SELECT s.supervisor_id, e.first_name, e.last_name, COUNT(r.route_id) AS
route_count FROM Supervisors s JOIN Employees e ON s.employee_id =
e.employee_id LEFT JOIN Routes r ON s.supervisor_id = r.supervisor_id
GROUP BY s.supervisor_id, e.first_name, e.last_name ORDER BY
route_count DESC;
```

supervisor_id	first_name	last_name	route_count
1	Andrew	Johnson	4
2	Sophia	Miller	0

h. Average Order Amount per Month (Aadil)

```
SELECT YEAR(order_date) AS year, MONTH(order_date) AS month,
AVG(total_amount) AS avg_order_amount FROM Orders GROUP BY year, month
ORDER BY year, month;
```

year	month	avg_order_amount
2023	6	7.730000
2023	7	10.538333

8. Database Optimization

4.1 Data Performance Tuning

Enhancing query response time through optimal resource utilization is the core goal of database performance tuning. The primary focus is to reduce user query retrieval time while maximizing resource efficiency.

4.2 Query Processing

Query processing involves parsing, optimizing, generating, and executing SQL statements to ensure swift results. Parsing includes segmenting SQL components and performing syntax, semantic, and shared pool checks. Execution determines the most suitable query execution strategy. Finally, fetching returns the query outcome based on a query-evaluation plan.

4.3 Query optimization

Query optimization aims to boost query performance using system resources and performance metrics. It entails identifying optimal techniques to improve efficiency. Strategies encompass selecting optimal execution methods and resource utilization to achieve improved performance.

Index implementation

Indexing speeds up column queries by creating pointers to where data is stored in a database. The following SQL statement generates an index called "idx_username" on the first_name andlast_name column in the "Customer" table:

SQL Statement-

```
CREATE INDEX idx_username
ON Customer (first_name, last_name);
```

View index:

table

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
CUSTOMERS	0	PRIMARY	1	customer_id	A	8	NULL	NULL		BTREE			YES	NULL
CUSTOMERS	0	email	1	email	A	8	NULL	NULL	YES	BTREE			YES	NULL
CUSTOMERS	1	idx_customer	1	first_name	A	8	NULL	NULL	YES	BTREE			YES	NULL
CUSTOMERS	1	idx_customer	2	last_name	A	8	NULL	NULL	YES	BTREE			YES	NULL

After creating the index, query performance improved, with data retrieval time decreasing from 0.04 to 0.018 seconds and CPU cost decreasing as well.

5.4 Hash Partitioning Implementation

Hash partitioning is primarily utilized to achieve a balanced data distribution across a specified partition count.

The subsequent SQL command generates a fresh table named Ride_Request_Hash_Tbl based on the existing Ride Request table. It employs hashing on the Ride_ID column and divides data into three partitions.

Hash partitioning is primarily used to achieve a balanced data distribution across a specified partition count.

Given below SQL query generates a new table name order_id_hash based on the existing Orders table. Hashing is done on the order_id_hash column and divides data into 3 partitions.

Query:

```
CREATE TABLE Order_hash (
    order_id_hash INT,
    customer_id INT,
    order_date DATE,
    total_amount DECIMAL(10, 2),
    -- Other columns...

    PRIMARY KEY (order_id_hash),
    CONSTRAINT fk_customer
    FOREIGN KEY (customer_id)
    REFERENCES Customer(customer_id)
)
PARTITION BY HASH (order_id_hash)
PARTITIONS 3;
```

After insertion of dummy data into table

order_id_hash	customer_id	order_date	total_amou...
3	103	2023-08-24	75.20
9	101	2023-08-25	300.75
1	101	2023-08-23	150.00
2	102	2023-08-23	200.50