Database Management System Final Project

CSC320

Database Management of Vee Coffee Shop

Presented by:

Roya F. Kais

Presented to:

Dr. Maroun Abi Assaf

May 2025

Analysis & Description:

The Vee Coffee Shop database project establishes a reliable and scalable data management system tailored to the operational needs of a modern coffeehouse. It centralizes critical information—ranging from sales and inventory to employee records and supplier data—into a single, structured database. This setup enhances data accuracy, improves workflow efficiency, and supports real-time access to essential metrics. By leveraging this database, Vee Coffee Shop can make informed decisions, reduce operational bottlenecks, and deliver a more consistent, data-driven customer experience

This Business may include:

- **Customer Management:**
  Vee collects and stores customer details through offering their loyalty programs which supports their relationship with the customer and their own personalized marketing. Those details include unique ID, phone number,and address
  Each customer can place multiple orders.

- **Product Cataloging:**
  These products are identified by unique product ID including the product's name, category, base pice.
  A product can be part of one order and be stocked in the coffee shop's depot and is supplied by the suppliers.

- **Branch Operations:**
  Considering Vee has 16 branches at the moment, each branch has a unique branch ID and it includes location, openning date and the size of the branch.
  Holds inventory
  One branch can employ multiple employees and stock many products, have receive deliveries, and have a manager who manages each branch.

- **Employee :**
  Identified uniquely by SSN, along with attributes such as first name and last name, position (part-time or full-time), salary, and hire date.

They support in-store operations as well as customer service.
Each employee belongs to one store.

- **Order Processing:**
  Orders are assigned by unique names, and include order date, total ammount, payment method, and status(on-hold). They represent purchases made either in shop or online.
  Each order is linked to one customer and contain many products

- **Inventory Management:**
  Tracks quantities of products available and last restock dates per product at each branch.
  It is a weak entitiy set that depends on branch ID

- **Supplier Coordination:**
  suppliers are identified by unique ID, name, phone number, email, and location.
  Vee coordinates with suppliers to acquire products.
  Each supplier can be responbsible for multiple deliveries and supplying many products.

- **Delivery Tracking:**
  Deliveries have a unique ID, including arriving date, delivery date and status(paid/not paid)
  It represents deliveries for Vee branches.
  Each delivery is linked to a supplier and is received by the branches.

Requirements

1. **Data Storage and Integrity:**

- Store comprehensive and accurate data on all key entities: customers, products, employees, branches, orders, inventory, suppliers, and delivery.
- Ensure data integrity through proper use of primary and foreign keys.

2. **Tracking and Monitoring:**

- Monitor branch-level inventory and product availability.
- Track customer orders, payment methods, and order-delivery status.
- Log employee data, supplier interactions, and delivery histories.

3. **Inventory and Sales Management:**

- Maintain real-time product availability across all branch locations.
- Link inventory to shipment records to track supply movements.
- Analyze sales patterns by product, branch, or customer group.

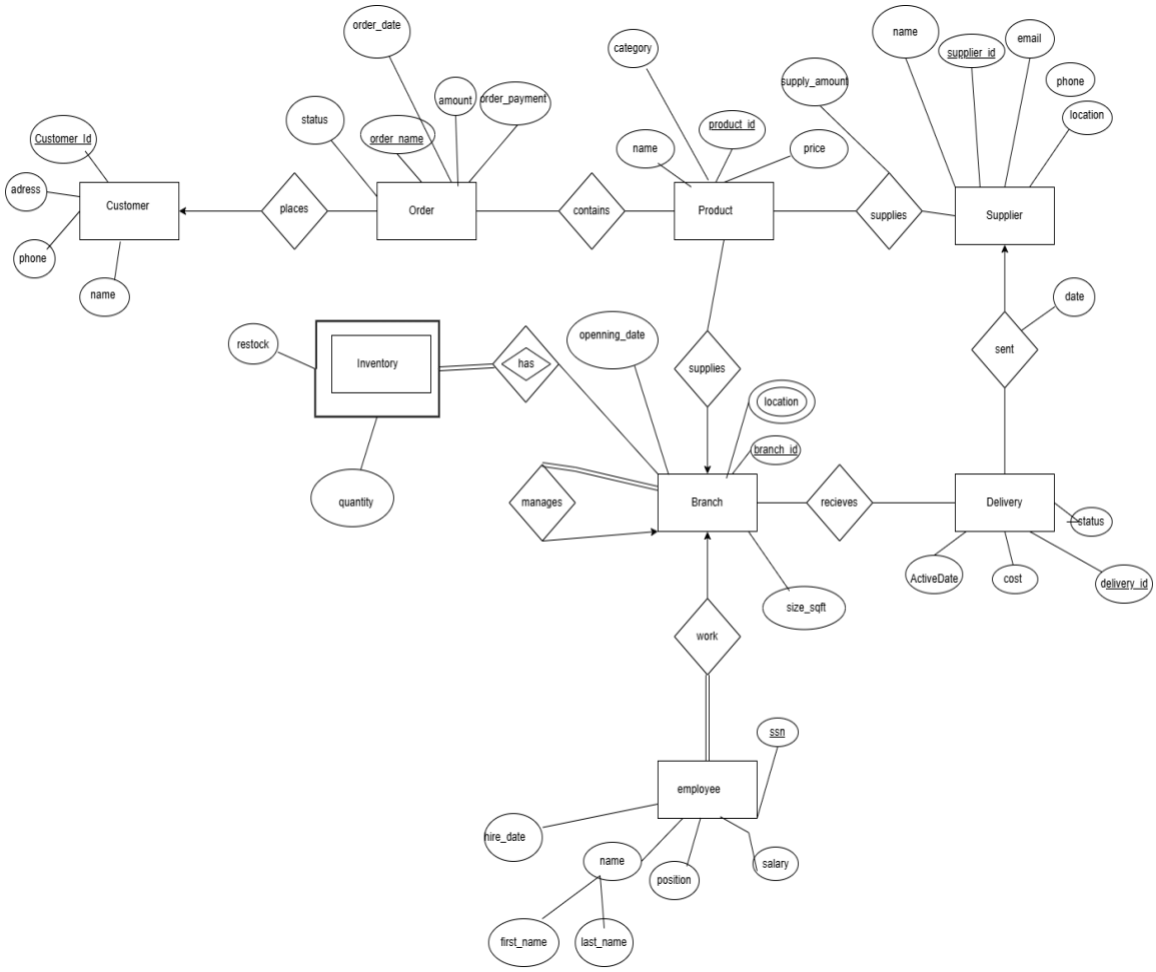
4. **Reporting and Analysis:**

- Generate insights on customer purchasing behavior, best-selling orders, and branch performance.
- Evaluate supplier reliability, employee staffing trends, and order volumes.

5. **Normalization:**

- Ensure the database is normalized to at least 3NF to avoid redundancy and ensure consistency across data records.

ER Diagram


order_date
amount
order_payment
status
order_name
Customer_Id
adress
Customer
phone
name
places
Order
contains
category
name
product_id
price
Product
supply_amount
supplies
name
supplier_id
email
phone
location
Supplier
restock
Inventory
has
openning_date
supplies
location
branch_id
quantity
manages
Branch
recieves
Delivery
date
sent
status
ActiveDate
cost
delivery_id
size_sqft
work
ssn
employee
hire_date
name
position
salary
first_name
last_name

Relational Schema

Customer (Customer_id:Integer, Name: String, Phone: integer, Address: String,)

Order (Order_Id: Integer, OrderDate :Date, Amount :Real, Status :String, PaymentMethod: Real, Customer_id*:Integer)

Product (Product_id: Integer, Name: String, Price: Real, Category: String, Order_Id*: Integer, store_id*: Integer)

Supplier(Supplier_id: Integer, Email: string, Name: String, phone:integer)

Supplies(product_id*: Integer, Supplier_Id*: Integer, supply_amount: Real)

Branch (Branch_id: Integer, Address: String, Size: Integer, Opening_date: Date, Manager_id*: integer)

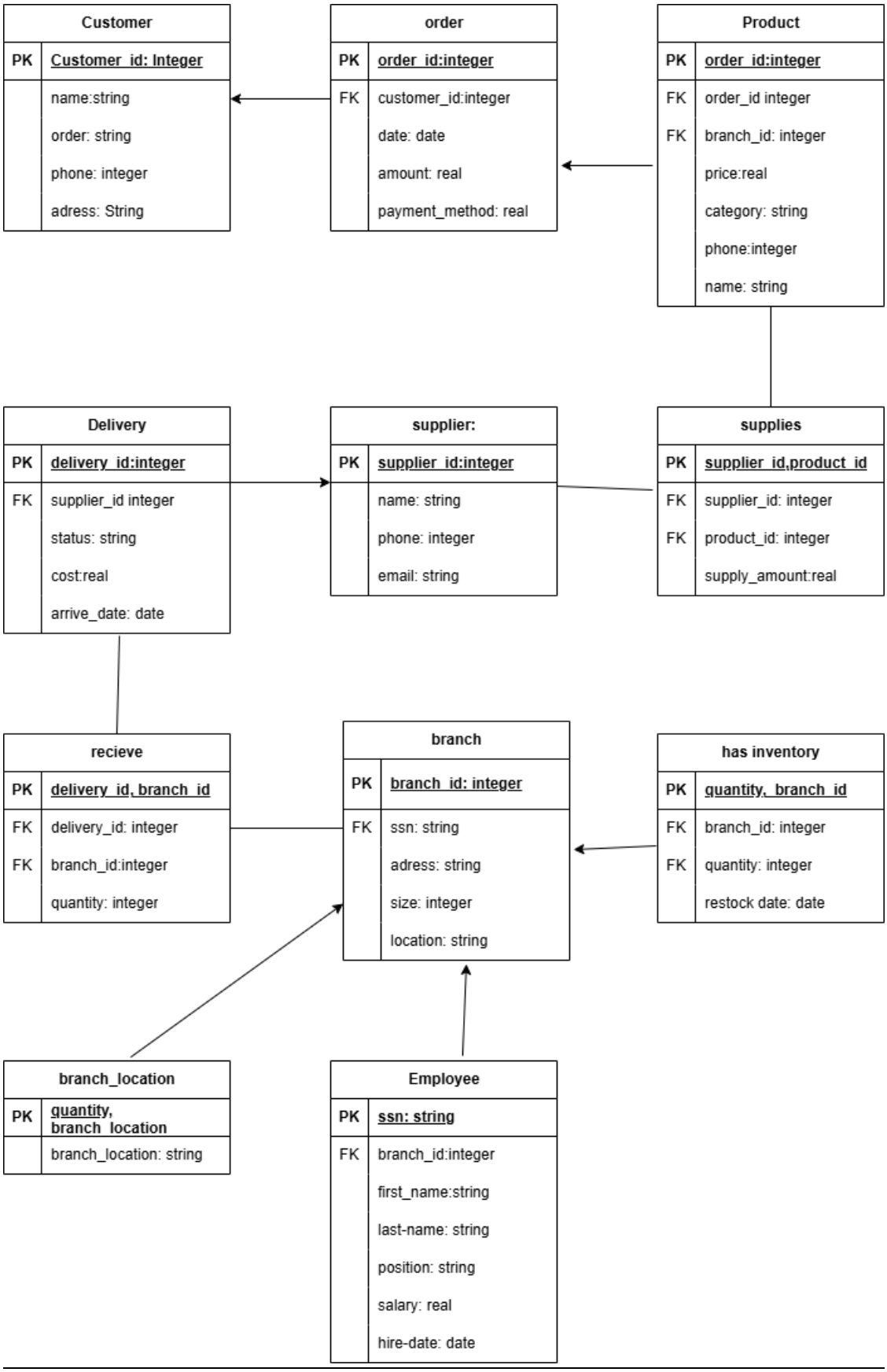
Branch_location(Branch_id*: Integer, Location:String)

Employee(SSN: String, firstName: String, lastName: String, Position: string, Salary: Real, Hire_date: Date, Branch_id*: Integer)

Inventory(quantity: Real, Branch_id*: Integer, Restock_date: Date)

Delivery(Delivery_id: Integer, Arrival_date: Date, Delivery_cost: Real, Status: String, Supplier_id*: Integer)

Receive (Delivery_id*: Integer, Branch_id*: Integer, Received_quantity:Real)


Customer
PK
Customer_id: Integer
name:string
order: string
phone: integer
adress: String
order
PK
order_id:integer
FK
customer_id:integer
date: date
amount: real
payment_method: real
Product
PK
order_id:integer
FK
order_id integer
FK
branch_id: integer
price:real
category: string
phone:integer
name: string
Delivery
PK
delivery_id:integer
FK
supplier_id integer
status: string
cost:real
arrive_date: date
supplier:
PK
supplier_id:integer
name: string
phone: integer
email: string
supplies
PK
supplier_id,product_id
FK
supplier_id: integer
FK
product_id: integer
supply_amount:real
recieve
PK
delivery_id, branch_id
FK
delivery_id: integer
FK
branch_id:integer
quantity: integer
branch
PK
branch_id: integer
FK
ssn: string
adress: string
size: integer
location: string
has inventory
PK
quantity, branch_id
FK
branch_id: integer
FK
quantity: integer
restock date: date
branch_location
PK
quantity,
branch_location
branch_location: string
Employee
PK
ssn: string
FK
branch_id:integer
first_name:string
last-name: string
position: string
salary: real
hire-date: date

Functional Dependency:

Customer(ID) → {name, address, phone number, }

OrderID → {Orderdate, amount, payment method , status, customer_ID}

Product(ID) → {name, price, category, order_Id, Branch_ID}

Supplier(ID) → {name, phone number, country, contact person, email}

(SupplierID, ProductID) → {Supply_amount}

BranchID → {address, size, opening date, SSN}

Employee(SSN) → {Fname, Lname, position, salary, hire date, storeID}

(BranchID, location) → {branchID, location}

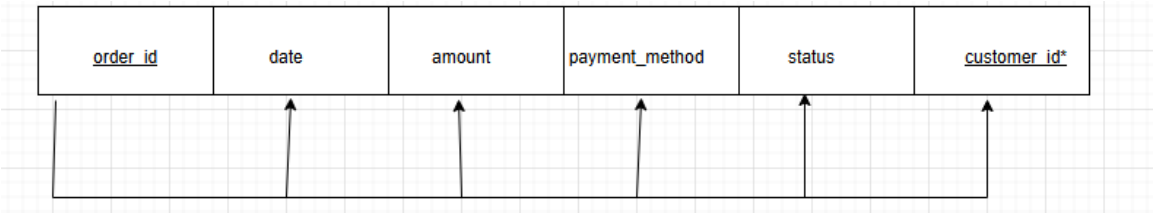
(BranchID, quantity) → {restock date}

Branch(ID) → {Cost, status, arrival date, supplierID}

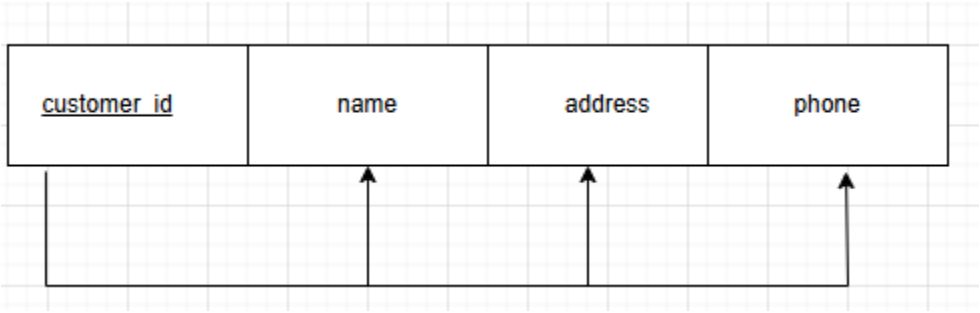
(ShipmentID, StoreID) → { received quantity}

Diagrams:

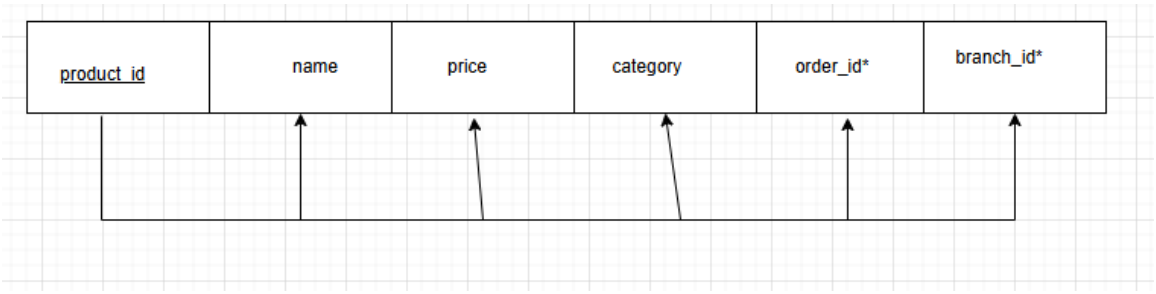
Owner:



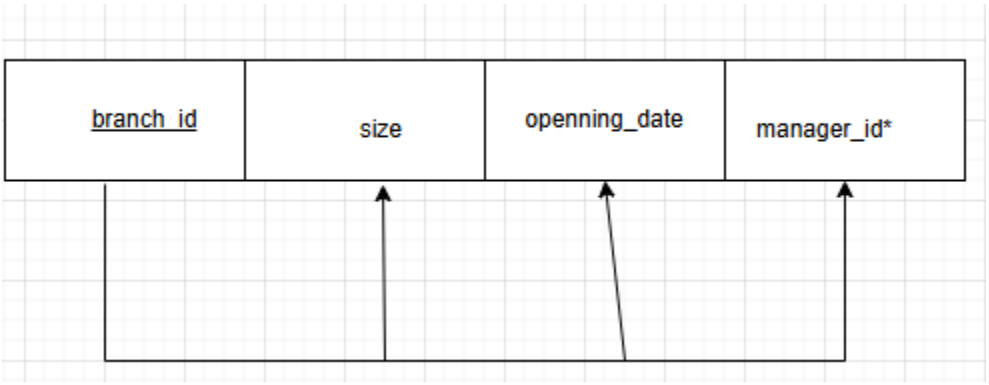
Customer:

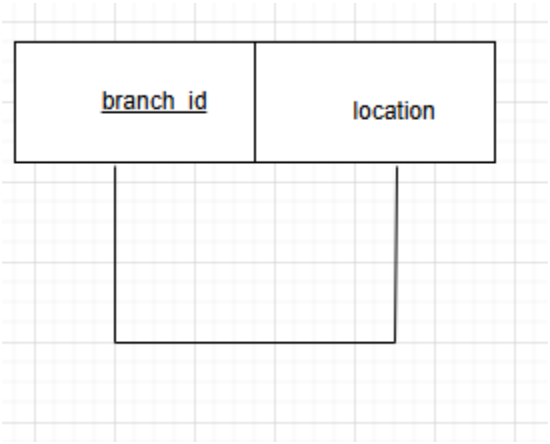


Product:

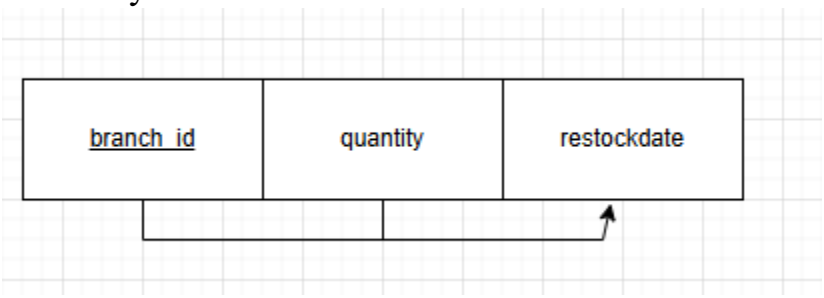


Branch:

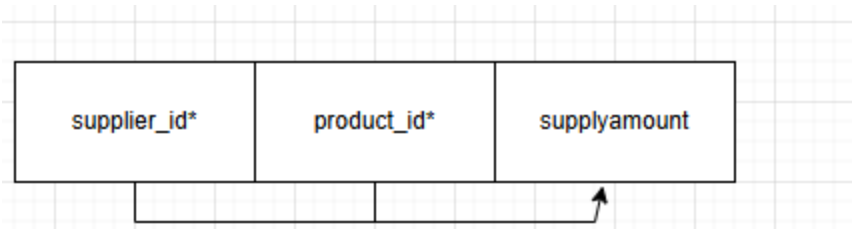


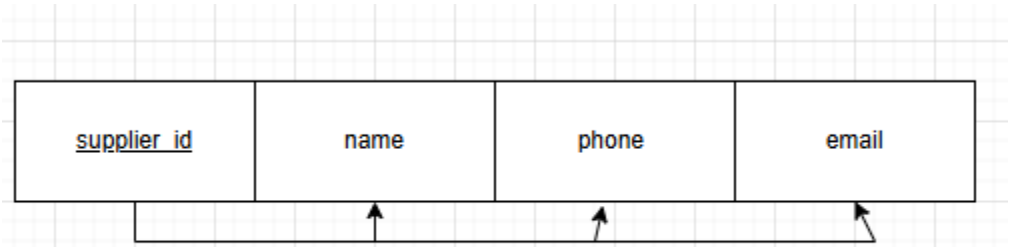


Inventory:

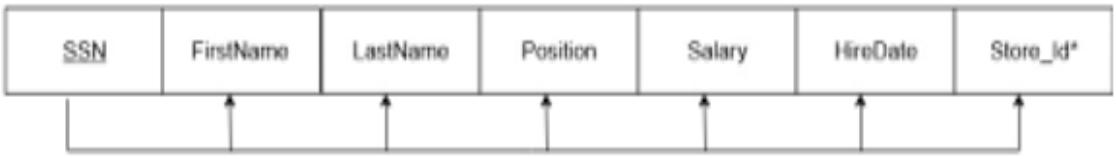


Supplier:





Employee:



Delivery:

Create table and Insert info:

```
CREATE SCHEMA IF NOT EXISTS Vdb
USE Vdb;

CREATE TABLE IF NOT EXISTS customer (
   customer_id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   phone VARCHAR(20),
```

```
    address TEXT
    );

    CREATE TABLE IF NOT EXISTS orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    order_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    total_amount DECIMAL(10,2) CHECK (total_amount >= 0),
    payment_method ENUM( 'Cash', 'Online Payment') NOT NULL,
    status ENUM( 'paid', 'on hold', 'Cancelled') DEFAULT 'paid',
    FOREIGN KEY (customer_id) REFERENCES
customer(customer_id)
);

CREATE TABLE IF NOT EXISTS supplier (
    supplier_id INT AUTO_INCREMENT PRIMARY KEY,
    Sname VARCHAR(100) NOT NULL,
    email VARCHAR(100),
    phone VARCHAR(20)
);

CREATE TABLE IF NOT EXISTS employee (
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    SSN VARCHAR(30) UNIQUE,
    branch_id INT,
    First_name VARCHAR(100) NOT NULL,
    Last_name VARCHAR(100) NOT NULL,
    position VARCHAR(50) NOT NULL,
    salary DECIMAL(10,2) CHECK (salary > 0),
    hire_date DATE
);
CREATE TABLE IF NOT EXISTS branch (
    branch_id INT AUTO_INCREMENT PRIMARY KEY,
    manager_id INT NOT NULL,
    size_sqft INT,
    opening_date DATE,
```

```
   FOREIGN KEY (manager_id) REFERENCES
employee(employee_id)
);
CREATE TABLE IF NOT EXISTS product (
   product_id INT AUTO_INCREMENT PRIMARY KEY,
   order_id INT NOT NULL,
   branch_id INT NOT NULL,
   name VARCHAR(100) NOT NULL,
   category VARCHAR(50),
   price DECIMAL(10,2) NOT NULL CHECK (price > 0),
   FOREIGN KEY(order_id) REFERENCES orders(order_id),
        FOREIGN KEY(branch_id) REFERENCES branch(branch_id)
);
CREATE TABLE supplies (
        product_id INT NOT NULL,
   supplier_id INT NOT NULL,
   supplyAmount DECIMAL(10,2),
   PRIMARY KEY(product_id,supplier_id),
        FOREIGN KEY (product_id) REFERENCES
product(product_id),
        FOREIGN KEY (supplier_id) REFERENCES
supplier(supplier_id)
);
CREATE TABLE branch_loction (
        branch_id INT NOT NULL,
   location VARCHAR(20),
   PRIMARY KEY(branch_id,location),
   FOREIGN KEY (branch_id) REFERENCES branch(branch_id)
);
CREATE TABLE IF NOT EXISTS delivery (
   delivery_id INT AUTO_INCREMENT PRIMARY KEY,
   supplier_id INT NOT NULL,
   arrival_date DATE,
   delivery_cost DECIMAL(10,2) CHECK (delivery_cost >= 0),
   status ENUM('Pending', 'In Transit', 'Delivered', 'Delayed')
DEFAULT 'Pending',
   FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id)
```

```
);
CREATE TABLE IF NOT EXISTS inventory (
   branch_id INT NOT NULL,
   quantity INT NOT NULL DEFAULT 0 CHECK (quantity >= 0),
   last_restock_date DATE,
   PRIMARY KEY (branch_id,quantity),
   FOREIGN KEY (branch_id) REFERENCES branch(branch_id)
ON DELETE CASCADE
);
CREATE TABLE IF NOT EXISTS receive_delivery (
   delivery_id INT NOT NULL,
   branch_id INT NOT NULL,
   received_quantity INT NOT NULL CHECK (received_quantity >
0),
   PRIMARY KEY (delivery_id, branch_id),
   FOREIGN KEY (delivery_id) REFERENCES
delivery(delivery_id),
   FOREIGN KEY (branch_id) REFERENCES branch(branch_id)
);
INSERT INTO customer (name, phone, address) VALUES
('Joelle Chedraoui',  '555-0101', 'jounieh'),
('Fady Ayoub',  '555-0102', 'Ghazir'),
('Rebecca Abi Rached', '555-0103', 'Jbeil');

INSERT INTO orders (customer_id, total_amount, payment_method,
status) VALUES
(1, 25.02, 'Online Payment', 'on hold'),
(2, 19.99, 'Online Payment', 'Paid'),
(3, 6.99, 'Cash', 'Paid');

INSERT INTO supplier (Sname, email, phone) VALUES
('Vee Main',  'main@veesupply.com', '555-0201'),
('Vee Zgharta',  'Zgharta@Veesupply.com', '555-0202'),
('Barista beans',  'barista@supply.com', '555-0203' );

INSERT INTO employee (employee_id, SSN, branch_id, First_name,
Last_name, position, salary, hire_date) VALUES
```

```
(1, '123-45-6789', NULL, 'Nadine', 'Rizk', 'Store Manager', 45000.00,
'2022-01-15'),
(2, '987-65-4321', NULL, 'Roya', 'Kais', 'barista', 38000.00, '2022-02-
20'),
(3, '456-78-9012', NULL, 'Houda', 'Rahme', 'Cashier', 32000.00,
'2022-03-10');

-- Insert stores referencing employee_id instead of SSN
INSERT INTO branch (manager_id, size_sqft, opening_date)
VALUES
(1, 8000, '2020-05-15'),
(2, 7500, '2021-03-20'),
(3, 9000, '2019-11-10');

-- Update employees with branch_ids
UPDATE employee e
JOIN branch s ON e.employee_id = s.manager_id
SET e.branch_id = s.branch_id
WHERE e.employee_id IN (SELECT manager_id FROM branch);

-- Add foreign key constraint
ALTER TABLE employee ADD CONSTRAINT fk_employee_store
FOREIGN KEY (branch_id) REFERENCES branch(branch_id);
INSERT INTO branch_loction (branch_id, location) VALUES
(1, 'Jounieh'),
(2, 'Jbeil'),
(3, 'Ammioun');

INSERT INTO product (order_id, branch_id, name, category, price)
VALUES
(1, 1, 'Espresso Beans', 'coffee', 19.99),
(1, 1, 'Lactose Free', 'milk', 29.99),
(2, 2, 'matcha', 'powder', 199.99),
(3, 3, 'eclaire', 'sweets', 10.99);
INSERT INTO supplies (product_id, supplier_id, supplyAmount)
VALUES
(1, 1, 12.00),
```

(2, 1, 35.00),
(3, 2, 120.00),
(4, 3, 25.00);
INSERT INTO delivery (supplier_id, arrival_date, delivery_cost, status) VALUES
(1, '2023-06-15', 50.00, 'Delivered'),
(2, '2023-06-20', 35.00, 'In Transit'),
(3, '2023-06-25', 40.00, 'Pending');
INSERT INTO inventory (branch_id, quantity, last_restock_date) VALUES
(1, 150, '2025-06-01'),
(2, 120, '2025-06-05'),
(3, 180, '2025-06-10');
INSERT INTO receive_delivery (delivery_id, branch_id, received_quantity) VALUES
(1, 1, 50),
(2, 2, 40),
(3, 3, 60);

10 business questions to answer:

```
154 •   Select* from customer;
```

Result Grid | Filter Rows: | Edit:

customer_id	name	phone	address
1	Joelle Chedraoui	555-0101	jounieh
2	Fady Ayoub	555-0102	Ghazir
3	Rebecca Abi Rached	555-0103	Jbeil
NULL	NULL	NULL	NULL

```
155
156 •   SELECT distinct name, address From customer;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

name	address
Joelle Chedraoui	jounieh
Fady Ayoub	Ghazir
Rebecca Abi Rached	Jbeil

```
157
158 •  SELECT* from orders;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

order_id	customer_id	order_date	total_amount	payment_method	status
1	1	2025-05-07 21:12:36	25.02	Online Payment	on hold
2	2	2025-05-07 21:12:36	19.99	Online Payment	paid
3	3	2025-05-07 21:12:36	6.99	Cash	paid
NULL	NULL	NULL	NULL	NULL	NULL

What is the total cost of deliveries for each supplier?

```
167 •  SELECT s.Sname, SUM(d.delivery_cost) AS total_delivery_cost
168    FROM delivery d
169    JOIN supplier s ON d.supplier_id = s.supplier_id
170    GROUP BY s.Sname;
171
172
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Sname	total_delivery_cost
Vee Main	50.00
Vee Zgharta	35.00
Barista beans	40.00

What is the average salary of employees by position?

```
171
172 •  SELECT position, AVG(salary) AS average_salary
173    FROM employee
174    GROUP BY position;
175
```

Result Grid | Filter Rows: | Export: | Wrap Cell

position	average_salary
Store Manager	45000.000000
barista	38000.000000
Cashier	32000.000000

How many orders are pending, and how much do they total?

```
175
176 •  SELECT COUNT(o.order_id) AS pending_orders, SUM(o.total_amount) AS total_pend
177    FROM orders o
178    WHERE o.status = 'on hold';
179
180
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pending_orders	total_pending
1	25.02

What are the total supply amounts for each product across all suppliers?

```
176
177 •  SELECT p.name, SUM(s.supplyAmount) AS total_supply
178    FROM product p
179    JOIN supplies s ON p.product_id = s.product_id
180    GROUP BY p.name;
181
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conten

name	total_supply
Espresso Beans	12.00
Lactose Free	35.00
matcha	120.00
eclaire	25.00

calculate the total sales per branch

```sql
SELECT b.branch_id, SUM(o.total_amount) AS total_sales
FROM orders o
JOIN product p ON o.order_id = p.order_id
JOIN branch b ON p.branch_id = b.branch_id
GROUP BY b.branch_id
LIMIT 0, 1000;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

branch_id	total_sales
1	50.04
2	19.99
3	6.99

```sql
select* from product;
```

Result Grid | Filter Rows: | Edit: | Export/Imp

product_id	order_id	branch_id	name	category	price
1	1	1	Espresso Beans	coffee	19.99
2	1	1	Lactose Free	milk	29.99
3	2	2	matcha	powder	199.99
4	3	3	eclaire	sweets	10.99
NULL	NULL	NULL	NULL	NULL	NULL

```
190
191 •  SELECT p.category, SUM(o.total_amount) AS total_sales
192    FROM orders o
193    JOIN product p ON o.order_id = p.order_id
194    GROUP BY p.category
195    ORDER BY total_sales DESC
196    LIMIT 0, 1000;
197
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category	total_sales
coffee	25.02
milk	25.02
powder	19.99
sweets	6.99

top-selling product categories based on the total sales