Database Systems and Programming

Neelkumar Patel
pateln93@students.rowan.edu
neelkumarpatel.com
Rowan University

March, 2023

Table of Contents

INTRODUCTION

Problem Domain: The database is for a retail company that sells various products to customers through their website. The database includes information about customers, products, orders, invoices, payments, employees, sales, categories, and suppliers.

Requirements:

- Customers should have a unique identifier, name, address, contact number, and email address.32
- Products should have a unique identifier, category, name, description, price, weight, UPS code, and inventory level.
- Orders should have a unique identifier, customer identifier, product identifier, order date, and quantity.
- Invoices should have a unique identifier, order identifier, payment identifier, discount identifier, customer identifier, invoice date, due date, total amount, discount amount, tax amount, paid amount, and payment status.
- Payments should have a unique identifier, order identifier, payment date, payment amount, and payment method.
- Employees should have a unique identifier, user name, password, name, and job title.
- Sales should have a unique identifier, customer identifier, employee identifier, date and time, and total amount.
- Categories should have a unique identifier, name, and description.
- Suppliers should have a unique identifier, name, address, city, state, country, contact information, and TIN number.
- Discounts should have a unique identifier, name, description, and discount amount.
- The inventory level of a product should be an integer that is required and cannot be negative.
- The weight of a product should be a real number that is required and cannot be negative.
- The quantity of an order should be an integer that is required and cannot be negative.
- The total amount of an order should be a real number that is required and cannot be negative.
- The total amount of an invoice should be a real number that is required and cannot be negative.
- The discount amount of an invoice should be a real number that is required and cannot be negative.
- The tax amount of an invoice should be a real number that is required and cannot be negative.
- The paid amount of an invoice should be a real number that is required and cannot be negative.
- The payment status of an invoice should be a text field that is required.

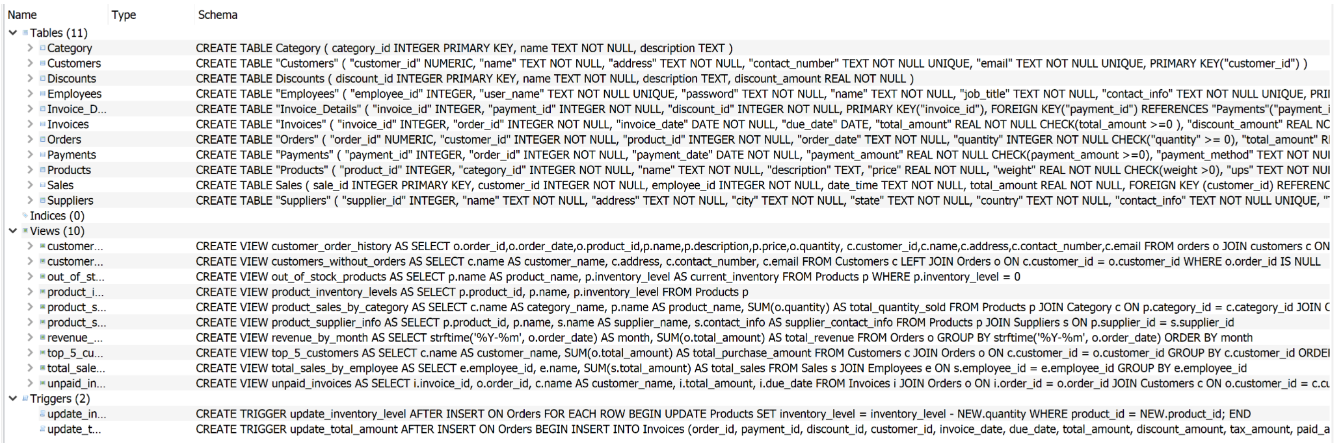
- The payment amount of a payment should be a real number that is required and cannot be negative.

Functional Dependencies:

- Products: product_id -> category_id, name, description, price, weight, ups, inventory_level, supplier_id
- Orders: order_id -> customer_id, product_id, order_date, quantity, total_amount
- Invoices: invoice_id -> order_id, payment_id, discount_id, customer_id, invoice_date, due_date, total_amount, discount_amount, tax_amount, paid_amount, payment_status
- Payments: payment_id -> order_id, payment_date, payment_amount, payment_method
- Employees: employee_id -> user_name, password, name, job_title, contact_info
- Sales: sale_id -> customer_id, employee_id, date_time, total_amount
- Category: category_id -> name, description
- Suppliers: supplier_id -> name, address, city, state, country, contact_info, TIN_number

Multivalued Dependencies:

- None identified in the given schema.



Name	Type	Schema
Tables (11)		
Category		CREATE TABLE Category (category_id INTEGER PRIMARY KEY, name TEXT NOT NULL, description TEXT)
Customers		CREATE TABLE "Customers" ("customer_id" NUMERIC, "name" TEXT NOT NULL, "address" TEXT NOT NULL, "contact_number" TEXT NOT NULL UNIQUE, "email" TEXT NOT NULL UNIQUE, PRIMARY KEY("customer_id"))
Discounts		CREATE TABLE Discounts (discount_id INTEGER PRIMARY KEY, name TEXT NOT NULL, description TEXT, discount_amount REAL NOT NULL)
Employees		CREATE TABLE "Employees" ("employee_id" INTEGER, "user_name" TEXT NOT NULL UNIQUE, "password" TEXT NOT NULL, "name" TEXT NOT NULL, "job_title" TEXT NOT NULL, "contact_info" TEXT NOT NULL UNIQUE, PRI
Invoice_D...		CREATE TABLE "Invoice_Details" ("invoice_id" INTEGER, "payment_id" INTEGER NOT NULL, "discount_id" INTEGER NOT NULL, PRIMARY KEY("invoice_id"), FOREIGN KEY("payment_id") REFERENCES "Payments"("payment_i
Invoices		CREATE TABLE "Invoices" ("invoice_id" INTEGER, "order_id" INTEGER NOT NULL, "invoice_date" DATE NOT NULL, "due_date" DATE, "total_amount" REAL NOT NULL CHECK(total_amount >=0), "discount_amount" REAL NC
Orders		CREATE TABLE "Orders" ("order_id" NUMERIC, "customer_id" INTEGER NOT NULL, "product_id" INTEGER NOT NULL, "order_date" TEXT NOT NULL, "quantity" INTEGER NOT NULL CHECK("quantity" >= 0), "total_amount" RI
Payments		CREATE TABLE "Payments" ("payment_id" INTEGER, "order_id" INTEGER NOT NULL, "payment_date" DATE NOT NULL, "payment_amount" REAL NOT NULL CHECK(payment_amount >=0), "payment_method" TEXT NOT NUI
Products		CREATE TABLE "Products" ("product_id" INTEGER, "category_id" INTEGER NOT NULL, "name" TEXT NOT NULL, "description" TEXT, "price" REAL NOT NULL, "weight" REAL NOT NULL CHECK(weight >0), "ups" TEXT NOT NUI
Sales		CREATE TABLE Sales (sale_id INTEGER PRIMARY KEY, customer_id INTEGER NOT NULL, employee_id INTEGER NOT NULL, date_time TEXT NOT NULL, total_amount REAL NOT NULL, FOREIGN KEY (customer_id) REFERENC
Suppliers		CREATE TABLE "Suppliers" ("supplier_id" INTEGER, "name" TEXT NOT NULL, "address" TEXT NOT NULL, "city" TEXT NOT NULL, "state" TEXT NOT NULL, "country" TEXT NOT NULL, "contact_info" TEXT NOT NULL UNIQUE, "
Indices (0)		
Views (10)		
customer...		CREATE VIEW customer_order_history AS SELECT o.order_id,o.order_date,o.product_id,p.name,p.description,p.price,o.quantity, c.customer_id,c.name,c.address,c.contact_number,c.email FROM orders o JOIN customers c ON
customer...		CREATE VIEW customers_without_orders AS SELECT c.name AS customer_name, c.address, c.contact_number, c.email FROM Customers c LEFT JOIN Orders o ON c.customer_id = o.customer_id WHERE o.order_id IS NULL
out_of_st...		CREATE VIEW out_of_stock_products AS SELECT p.name AS product_name, p.inventory_level AS current_inventory FROM Products p WHERE p.inventory_level = 0
product_i...		CREATE VIEW product_inventory_levels AS SELECT p.product_id, p.name, p.inventory_level FROM Products p
product_s...		CREATE VIEW product_sales_by_category AS SELECT c.name AS category_name, p.name AS product_name, SUM(o.quantity) AS total_quantity_sold FROM Products p JOIN Category c ON p.category_id = c.category_id JOIN C
product_s...		CREATE VIEW product_supplier_info AS SELECT p.product_id, p.name, s.name AS supplier_name, s.contact_info AS supplier_contact_info FROM Products p JOIN Suppliers s ON p.supplier_id = s.supplier_id
revenue_...		CREATE VIEW revenue_by_month AS SELECT strftime('%Y-%m', o.order_date) AS month, SUM(o.total_amount) AS total_revenue FROM Orders o GROUP BY strftime('%Y-%m', o.order_date) ORDER BY month
top_5_cu...		CREATE VIEW top_5_customers AS SELECT c.name AS customer_name, SUM(o.total_amount) AS total_purchase_amount FROM Customers c JOIN Orders o ON c.customer_id = o.customer_id GROUP BY c.customer_id ORDEI
total_sale...		CREATE VIEW total_sales_by_employee AS SELECT e.employee_id, e.name, SUM(s.total_amount) AS total_sales FROM Sales s JOIN Employees e ON s.employee_id = e.employee_id GROUP BY e.employee_id
unpaid_in...		CREATE VIEW unpaid_invoices AS SELECT i.invoice_id, o.order_id, c.name AS customer_name, i.total_amount, i.due_date FROM Invoices i JOIN Orders o ON i.order_id = o.order_id JOIN Customers c ON o.customer_id = c.cu
Triggers (2)		
update_in...		CREATE TRIGGER update_inventory_level AFTER INSERT ON Orders FOR EACH ROW BEGIN UPDATE Products SET inventory_level = inventory_level - NEW.quantity WHERE product_id = NEW.product_id; END
update_t...		CREATE TRIGGER update_total_amount AFTER INSERT ON Orders BEGIN INSERT INTO Invoices (order_id, payment_id, discount_id, customer_id, invoice_date, due_date, total_amount, discount_amount, tax_amount, paid_a

Figure 1.1 -database_structure

http://neelkumarpatel.com/Projects/Advance_database/Mid_project/Project_Report.pdf

ER Diagram

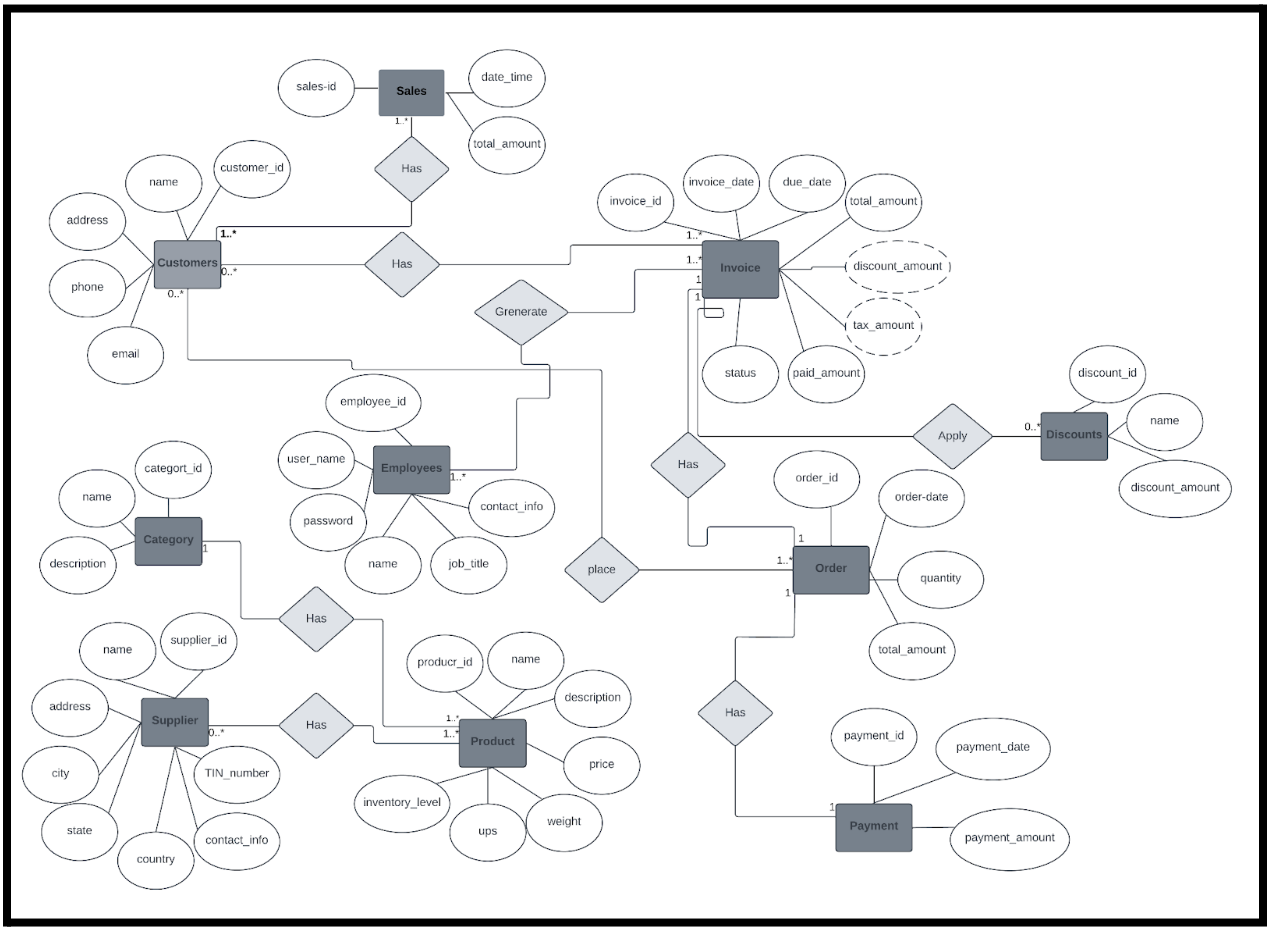
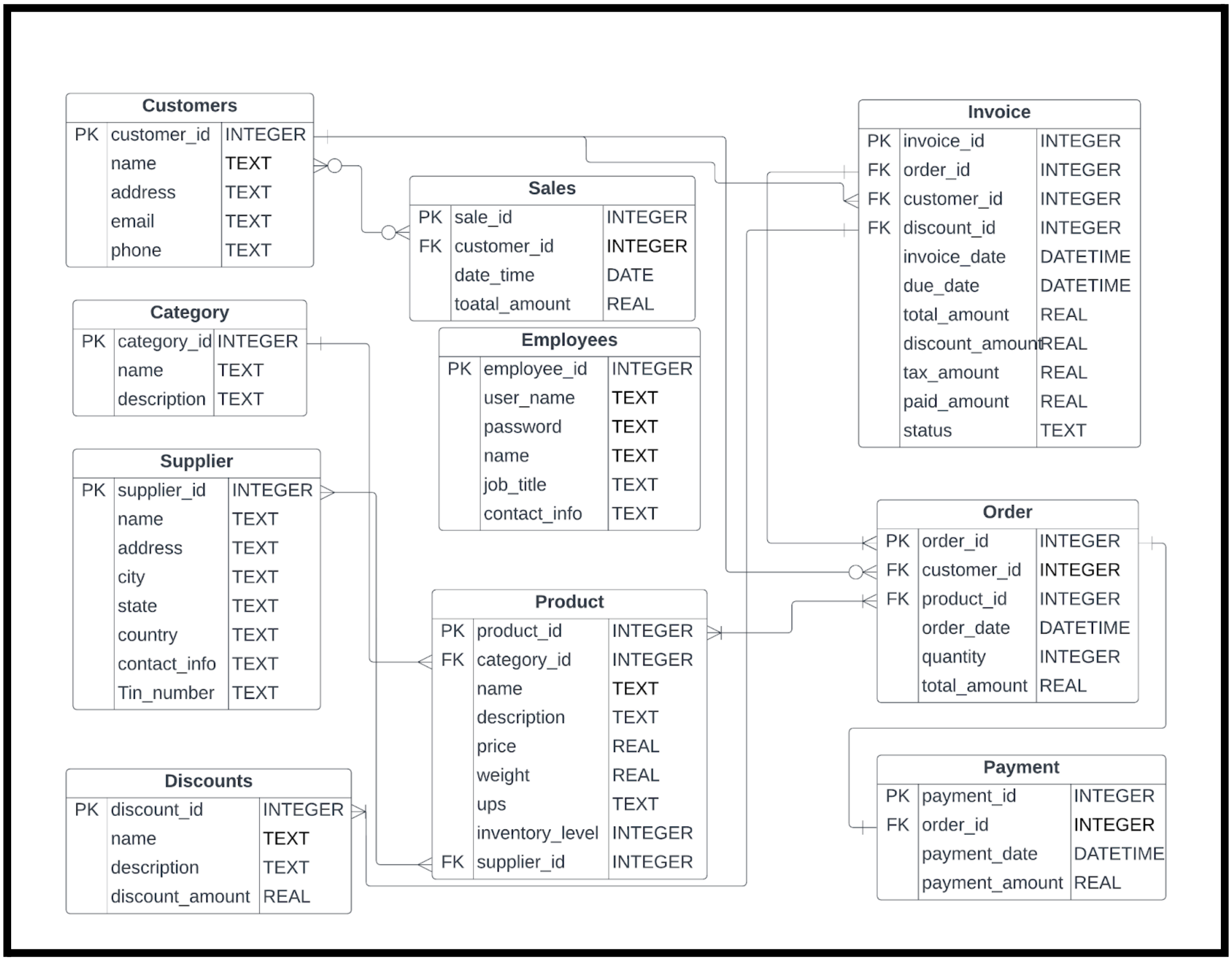


figure2.1

http://neelkumarpatel.com/Projects/Advance_database/Mid_project/er_diagram.pdf

Relational Database Design



Customers

PK	customer_id	INTEGER
	name	TEXT
	address	TEXT
	email	TEXT
	phone	TEXT

Sales

PK	sale_id	INTEGER
FK	customer_id	INTEGER
	date_time	DATE
	toatal_amount	REAL

Invoice

PK	invoice_id	INTEGER
FK	order_id	INTEGER
FK	customer_id	INTEGER
FK	discount_id	INTEGER
	invoice_date	DATETIME
	due_date	DATETIME
	total_amount	REAL
	discount_amount	REAL
	tax_amount	REAL
	paid_amount	REAL
	status	TEXT

Category

PK	category_id	INTEGER
	name	TEXT
	description	TEXT

Employees

PK	employee_id	INTEGER
	user_name	TEXT
	password	TEXT
	name	TEXT
	job_title	TEXT
	contact_info	TEXT

Supplier

PK	supplier_id	INTEGER
	name	TEXT
	address	TEXT
	city	TEXT
	state	TEXT
	country	TEXT
	contact_info	TEXT
	Tin_number	TEXT

Product

PK	product_id	INTEGER
FK	category_id	INTEGER
	name	TEXT
	description	TEXT
	price	REAL
	weight	REAL
	ups	TEXT
	inventory_level	INTEGER
FK	supplier_id	INTEGER

Order

PK	order_id	INTEGER
FK	customer_id	INTEGER
FK	product_id	INTEGER
	order_date	DATETIME
	quantity	INTEGER
	total_amount	REAL

Discounts

PK	discount_id	INTEGER
	name	TEXT
	description	TEXT
	discount_amount	REAL

Payment

PK	payment_id	INTEGER
FK	order_id	INTEGER
	payment_date	DATETIME
	payment_amount	REAL

Figure3.1

http://neelkumarpatel.com/Projects/Advance_database/Mid_project/relational_model.pdf

Convert Database in 3NF

Step 1: Analyze the functional dependencies
Identify all the functional dependencies between attributes in each table. Determine the andidate keys for each table

Step 2: Convert tables to 1NF
Ensure that each table has a primary key
Remove any repeating groups
Ensure that each column contains atomic values

Step 3: Convert tables to 2NF
Remove any partial dependencies by splitting tables into separate tables

Step 4: Convert tables to 3NF
Remove any transitive dependencies by splitting tables into separate tables
Step 1: Analyze the functional dependencies

Step 1: Analyze the functional dependencies

Customers table:
customer_id -> name, address, contact_number, email (customer_id is the candidate key)
Products table:
product_id -> category_id, name, description, price, weight, ups, inventory_level, supplier_id (product_id is the candidate key)
Orders table:
order_id -> customer_id, product_id, order_date, quantity, total_amount (order_id is the candidate key)
Invoices table:
invoice_id -> order_id, payment_id, discount_id, customer_id, invoice_date, due_date, total_amount, discount_amount, tax_amount, paid_amount, payment_status (invoice_id is the candidate key)
Payments table:
payment_id -> order_id, payment_date, payment_amount, payment_method (payment_id is the candidate key)
Employees table:
employee_id -> user_name, password, name, job_title, contact_info (employee_id is the candidate key)
Sales table:
sale_id -> customer_id, employee_id, date_time, total_amount (sale_id is the candidate key)
Category table:

category_id -> name, description (category_id is the candidate key)

Suppliers table:

supplier_id -> name, address, city, state, country, contact_info, TIN_number (supplier_id is the candidate key)

Discounts table:

discount_id -> name, description, discount_amount (discount_id is the candidate key)

Step 2: Convert tables to 1NF

All tables already have a primary key, so this step is complete.

Step 3: Convert tables to 2NF

Customers table:

No partial dependencies are present, so this table is already in 2NF.

Products table:

No partial dependencies are present, so this table is already in 2NF.

Orders table:

No partial dependencies are present, so this table is already in 2NF.

Invoices table:

payment_id, discount_id, customer_id, and order_id are functionally dependent on invoice_id, creating partial dependencies.

We can split the table into two:

>Invoices table: invoice_id, order_id, invoice_date, due_date, total_amount, discount_amount, tax_amount, paid_amount, payment_status

>Invoice_Details table: invoice_id, payment_id, discount_id, customer_id

Payments table:

No partial dependencies are present, so this table is already in 2NF.

Employees table:

No partial dependencies are present, so this table is already in 2NF.

Sales table:

No partial dependencies are present, so this table is already in 2NF.

Category table:

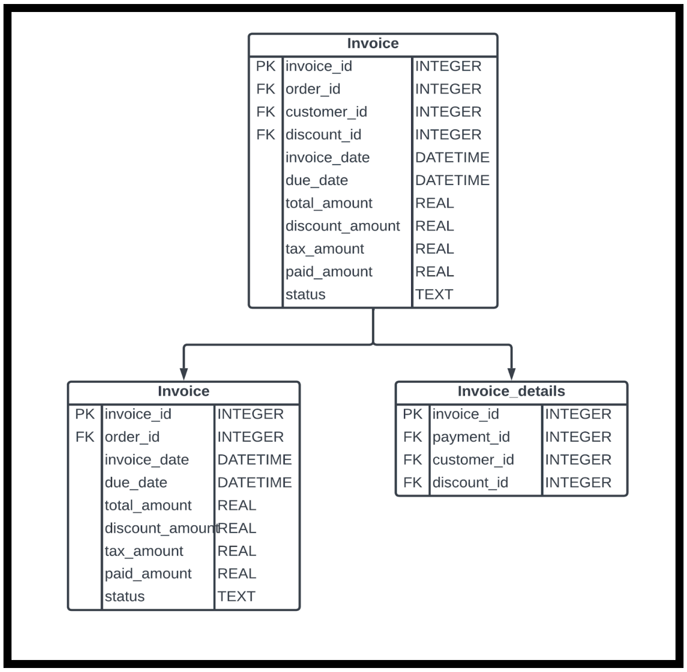
No partial dependencies are present, so this table is already in 2NF.

Suppliers table:

No partial dependencies are present, so this table is already in 2NF.

Discounts table:

No partial dependencies are present, so this table is already in 2NF.



Invoice		
PK	invoice_id	INTEGER
FK	order_id	INTEGER
FK	customer_id	INTEGER
FK	discount_id	INTEGER
	invoice_date	DATETIME
	due_date	DATETIME
	total_amount	REAL
	discount_amount	REAL
	tax_amount	REAL
	paid_amount	REAL
	status	TEXT

Invoice		
PK	invoice_id	INTEGER
FK	order_id	INTEGER
	invoice_date	DATETIME
	due_date	DATETIME
	total_amount	REAL
	discount_amount	REAL
	tax_amount	REAL
	paid_amount	REAL
	status	TEXT

Invoice_details		
PK	invoice_id	INTEGER
FK	payment_id	INTEGER
FK	customer_id	INTEGER
FK	discount_id	INTEGER

figure4.1

Step 4: Convert tables to 3NF

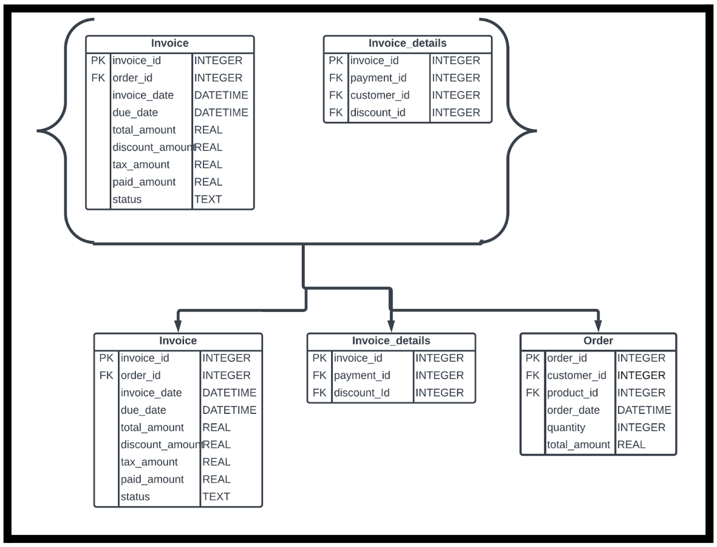
Invoices table:

The Invoice_Details table still contains a transitive dependency, as customer_id is functionally dependent on invoice_id through order_id. To remove this transitive dependency, we can split the table further into three tables:

Invoices table: invoice_id, order_id, invoice_date, due_date, total_amount, discount_amount, tax_amount, paid_amount, payment_status

Orders table: order_id, customer_id, product_id, order_date, quantity, total_amount

Invoice_Details table: invoice_id, payment_id, discount_id



Invoice		
PK	invoice_id	INTEGER
FK	order_id	INTEGER
	invoice_date	DATETIME
	due_date	DATETIME
	total_amount	REAL
	discount_amount	REAL
	tax_amount	REAL
	paid_amount	REAL
	status	TEXT

Invoice_details		
PK	invoice_id	INTEGER
FK	payment_id	INTEGER
FK	customer_id	INTEGER
FK	discount_id	INTEGER

Invoice		
PK	invoice_id	INTEGER
FK	order_id	INTEGER
	invoice_date	DATETIME
	due_date	DATETIME
	total_amount	REAL
	discount_amount	REAL
	tax_amount	REAL
	paid_amount	REAL
	status	TEXT

Invoice_details		
PK	invoice_id	INTEGER
FK	payment_id	INTEGER
FK	discount_Id	INTEGER

Order		
PK	order_id	INTEGER
FK	customer_id	INTEGER
FK	product_id	INTEGER
	order_date	DATETIME
	quantity	INTEGER
	total_amount	REAL

figure 4.2

Final Schema After performing Normalization:

Customers table:
customer_id -> name, address, contact_number, email (customer_id is the candidate key)

Products table:
product_id -> category_id, name, description, price, weight, ups, inventory_level, supplier_id (product_id is the candidate key)

Orders table:
order_id -> customer_id, product_id, order_date, quantity, total_amount (order_id is the candidate key)

Invoices table:
invoice_id -> order_id, invoice_date, due_date, total_amount, discount_amount, tax_amount, paid_amount, payment_status (invoice_id is the candidate key)

Invoice_Details table:
invoice_id, payment_id, discount_id

Payments table:
payment_id -> order_id, payment_date, payment_amount, payment_method (payment_id is the candidate key)

Employees table:
employee_id -> user_name, password, name, job_title, contact_info (employee_id is the candidate key)

Sales table:
sale_id -> customer_id, employee_id, date_time, total_amount (sale_id is the candidate key)

Category table:
category_id -> name, description (category_id is the candidate key)

Suppliers table:
supplier_id -> name, address, city, state, country, contact_info, TIN_number (supplier_id is the candidate key)

Discounts table:
discount_id -> name, description, discount_amount (discount_id is the candidate key)

Final Relational Diagram

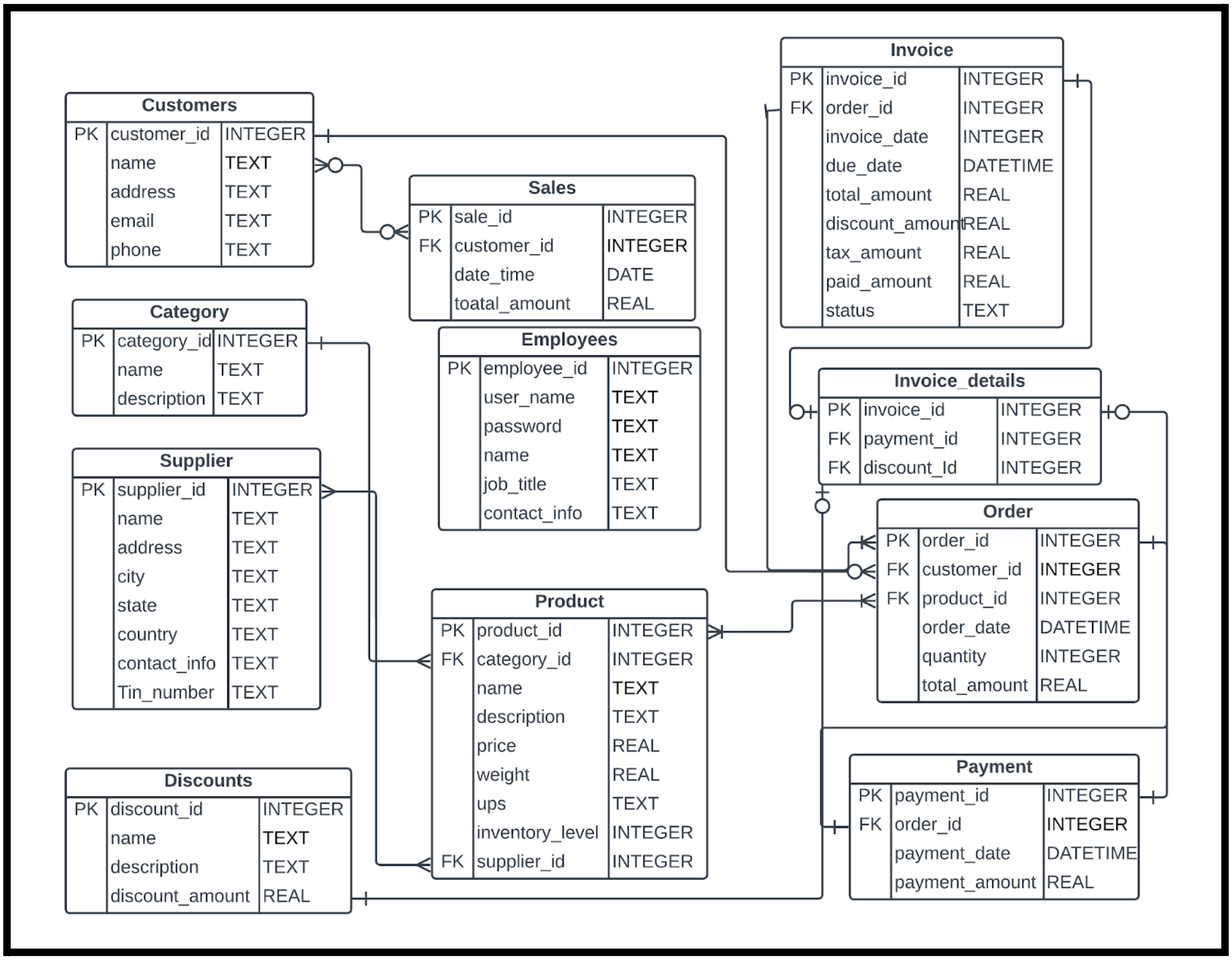


Figure4.3- relational model

http://neelkumarpatel.com/Projects/Advance_database/Mid_project/relational_model.pdf

SQL STATEMENTS

Create Table Statements:

http://neelkumarpatel.com/Projects/Advance_database/Mid_project/Create_Statement.sql

1) create the Customers table

```
CREATE TABLE "Customers" (
        "customer_id" NUMERIC,
        "name"TEXT NOT NULL,
        "address"       TEXT NOT NULL,
        "contact_number"        TEXT NOT NULL UNIQUE,
        "email"         TEXT NOT NULL UNIQUE,
        PRIMARY KEY("customer_id")
);
```

2) create the Products table

```
CREATE TABLE "Products" (
        "product_id"    INTEGER,
        "category_id"   INTEGER NOT NULL,
        "name"TEXT NOT NULL,
        "description"   TEXT,
        "price" REAL NOT NULL,
        "weight"        REAL NOT NULL CHECK(weight >0),
        "ups"   TEXT NOT NULL,
        "inventory_level"       NUMERIC NOT NULL CHECK("inventory_level" >= 0),
        "supplier_id"   INTEGER NOT NULL,
        PRIMARY KEY("product_id"),
        FOREIGN KEY("category_id") REFERENCES "Category"("category_id"),
        FOREIGN KEY("supplier_id") REFERENCES "Suppliers"("supplier_id")
);
```

3)create the Orders table

```
CREATE TABLE "Orders" (
        "order_id"      NUMERIC,
        "customer_id" INTEGER NOT NULL,
        "product_id"    INTEGER NOT NULL,
        "order_date"    TEXT NOT NULL,
        "quantity"      INTEGER NOT NULL CHECK("quantity" >= 0),
        "total_amount"          REAL NOT NULL CHECK(total_amount >=0),
	FOREIGN KEY("customer_id") REFERENCES "Customers"("customer_id"),
	FOREIGN KEY("product_id") REFERENCES "Products"("product_id"),
	PRIMARY KEY("order_id")
);
```

4) create the Invoices table

```
CREATE TABLE "Invoices" (
	"invoice_id"	INTEGER,
	"order_id"	INTEGER NOT NULL,
	"invoice_date"DATE NOT NULL,
	"due_date"	DATE,
	"total_amount"		REAL NOT NULL CHECK(total_amount >=0 ),
	"discount_amount"	REAL NOT NULL CHECK(discount_amount >=0),
	"tax_amount"	REAL NOT NULL CHECK(tax_amount >= 0),
	"paid_amount"		REAL NOT NULL CHECK(paid_amount >= 0),
	"payment_status"	TEXT NOT NULL,
	FOREIGN KEY("order_id") REFERENCES "Orders"("order_id"),
	PRIMARY KEY("invoice_id")
);
```

5)create invoice_details

```
CREATE TABLE "Invoice_Details" (
	"invoice_id"	INTEGER,
	"payment_id"	INTEGER NOT NULL,
	"discount_id"	INTEGER NOT NULL,
	PRIMARY KEY("invoice_id"),
	FOREIGN KEY("payment_id") REFERENCES "Payments"("payment_id"),
	FOREIGN KEY("discount_id") REFERENCES "Discounts"("discount_id"),
	FOREIGN KEY("invoice_id") REFERENCES "Invoices"("invoice_id")
);
```

6) create the Payments table

```
CREATE TABLE "Payments" (
	"payment_id"	INTEGER,
	"order_id"	INTEGER NOT NULL,
	"payment_date"		DATE NOT NULL,
	"payment_amount"	REAL NOT NULL CHECK(payment_amount >=0),
	"payment_method"	TEXT NOT NULL,
	FOREIGN KEY("order_id") REFERENCES "Orders"("order_id"),
	PRIMARY KEY("payment_id")
);
```

7) create the Employees table

```
CREATE TABLE "Employees" (
        "employee_id"INTEGER,
        "user_name"   TEXT NOT NULL UNIQUE,
        "password"    TEXT NOT NULL,
        "name"TEXT NOT NULL,
        "job_title"   TEXT NOT NULL,
        "contact_info" TEXT NOT NULL UNIQUE,
        PRIMARY KEY("employee_id")
);
```

8) create the Sales table

```
CREATE TABLE Sales (
  sale_id INTEGER PRIMARY KEY,
  customer_id INTEGER NOT NULL,
  employee_id INTEGER NOT NULL,
  date_time TEXT NOT NULL,
  total_amount REAL NOT NULL,
  FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
  FOREIGN KEY (employee_id) REFERENCES Employees(employee_id)
);
```

9) create the Category table

```
CREATE TABLE Category (
  category_id INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  description TEXT
);
```

10) create the Suppliers table

```
CREATE TABLE "Suppliers" (
        "supplier_id"   INTEGER,
        "name"TEXT NOT NULL,
        "address"     TEXT NOT NULL,
        "city"   TEXT NOT NULL,
        "state"  TEXT NOT NULL,
        "country"     TEXT NOT NULL,
	"contact_info" TEXT NOT NULL UNIQUE,
	"TIN_number"		TEXT NOT NULL UNIQUE,
	PRIMARY KEY("supplier_id")
);
```

11) create Discount table:

```
CREATE TABLE Discounts (
  discount_id INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  description TEXT,
  discount_amount REAL NOT NULL
);
```

Insert Statements

http://neelkumarpatel.com/Projects/Advance_database/Mid_project/Insert_Statements.sql

```
INSERT INTO Customers (customer_id,name, address, contact_number, email)
VALUES
			 (1, 'John Doe', '123 Main St', '555-1234', 'john.doe@example.com'),
			(2,'Jane Doe', '456 Park Ave', '555-555-5678', 'jane.doe@email.com'),
			(3,'Bob Johnson', '789 Elm St', '555-555-9012', 'bob.johnson@email.com'),
			(4,'Sarah Lee', '321 Maple Ave', '555-555-3456', 'sarah.lee@email.com'),
			(5,'David Chen', '654 Pine St', '555-555-7890', 'david.chen@email.com'),
			(6,'Maria Garcia', '987 Oak Ave', '555-555-2345', 'maria.garcia@email.com'),
			(7,'Michael Brown', '321 Elm St', '555-555-6789', 'michael.brown@email.com'),
			(8,'Laura Davis', '654 Main St', '555-555-0123', 'laura.davis@email.com'),
			(9,'Peter Kim', '789 Maple Ave', '555-555-4567', 'peter.kim@email.com'),
			(10,'Amanda Lee', '123 Pine St', '555-555-8901', 'amanda.lee@email.com');

INSERT INTO Suppliers (supplier_id, name, address, city, state, country, contact_info,
TIN_number) VALUES
		(1, 'ABC Suppliers', '123 Main St', 'Anytown', 'CA', 'USA', '555-1234', '123-45-6789'),
		(2, 'XYZ Company', '456 Elm St', 'Anycity', 'NY', 'USA', '555-5678', '987-65-4321'),
		(3, 'Acme Corporation', '789 Oak St', 'Anystate', 'TX', 'USA', '555-9012', '543-21-6789'),
		(4, 'Best Foods', '321 Maple St', 'Anycity', 'FL', 'USA', '555-3456', '876-54-3210'),
		(5, 'Global Imports', '654 Pine St', 'Anystate', 'CA', 'USA', '555-7890', '210-98-7654'),
		(6, 'Harvest Farms', '987 Cedar St', 'Anycity', 'NY', 'USA', '555-2345', '654-32-1098'),
		(7, 'Natures Bounty', '876 Birch St', 'Anystate', 'TX', 'USA', '555-6789', '890-12-3456'),
		(8, 'Organic Foods', '543 Cherry St', 'Anytown', 'CA', 'USA', '555-0123', '456-78-9012'),
        (9, 'Pure Protein', '210 Walnut St', 'Anycity', 'NY', 'USA', '555-4567', '789-01-2345'),
        (10, 'Super Supplements', '999 Chestnut St', 'Anystate', 'FL', 'USA', '555-8901',
        321-09-8765');
INSERT INTO Category (category_id, name, description)
VALUES
        (1, 'Grains', 'Cereal crops that are grown for their edible seeds'),
        (2, 'Pulses', 'Edible seeds of plants in the legume family'),
        (3, 'Oil Seeds', 'Seeds that are primarily grown for oil extraction'),
        (4, 'Flour', 'Ground grains used for baking and cooking'),
        (5, 'Others', 'Other food products that do not fit into the above categories');

INSERT INTO Products (product_id, category_id, name, description, price, weight, ups,
inventory_level, supplier_id)
VALUES
        (1, 1, 'Wheat', 'A cereal grain used to make flour for bread', 5.99, 2.0, '123456789012',
        100, 1),
        (2, 2, 'Lentils', 'A type of pulse used in soups and stews', 3.49, 1.5, '234567890123', 75,
        2),
        (3, 3, 'Soybeans', 'A type of oilseed used to make soybean oil and other products', 10.99,
        3.0, '345678901234', 50, 3),
        (4, 4, 'All-Purpose Flour', 'A versatile flour used for baking', 2.99, 2.5, '456789012345',
        200, 4),
        (5, 1, 'Rice', 'A cereal grain used as a staple food', 4.99, 2.0, '567890123456', 150, 5),
        (6, 2, 'Chickpeas', 'A type of pulse used in Middle Eastern and Indian cuisine', 4.49, 1.5,
        '678901234567', 100, 6),
        (7, 3, 'Canola Seeds', 'A type of oilseed used to make canola oil', 8.99, 3.0,
        '789012345678', 50, 7),
        (8, 4, 'Bread Flour', 'A high-protein flour used for making bread', 3.49, 2.5,
        '890123456789', 100, 8),
        (9, 1, 'Barley', 'A cereal grain used for brewing and as a food source', 6.99, 2.0,
        '901234567890', 75, 9),
        (10, 5, 'Granola', 'A breakfast food consisting of rolled oats, nuts, and dried fruit', 7.99,
        1.0, '012345678901', 100, 10);

INSERT INTO Discounts (discount_id, name, description, discount_amount)
VALUES
        (11,'Regular Price','no discount at this time',0),
        (1, 'New Customer Discount', 'Get 10% off your first purchase', 0.1),
        (2, 'Holiday Sale', '25% off all items during the month of December', 0.25),
        (3, 'Clearance', 'Up to 50% off select items', 0.5),
        (4, 'Bulk Discount', 'Buy 10 or more of the same item and get 15% off', 0.15),
        (5, 'Membership Discount', '10% off for members', 0.1),
        (6, 'Student Discount', '15% off with valid student ID', 0.15),
        (7, 'Military Discount', '15% off with valid military ID', 0.15),
        (8, 'Senior Discount', '10% off for customers over 65', 0.1),
        (9, 'Birthday Discount', '20% off on your birthday', 0.2),
        (10, 'Referral Discount', 'Get 10% off for every friend you refer', 0.1);


INSERT INTO Employees (employee_id, user_name, password, name, job_title, contact_info)
VALUES
        (1, 'johndoe', 'password123', 'John Doe', 'Manager', '555-1234'),
        (2, 'janedoe', 'password456', 'Jane Doe', 'Sales Associate', '555-5678'),
        (3, 'bobsmith', 'password789', 'Bob Smith', 'Shipping Coordinator', '555-9012'),
        (4, 'sarahjones', 'passwordabc', 'Sarah Jones', 'Customer Service Representative',
        '555-3456'),
        (5, 'mikebrown', 'passworddef', 'Mike Brown', 'IT Specialist', '555-7890');

INSERT INTO Orders (order_id, customer_id, product_id, order_date, quantity, total_amount)
VALUES
        (1, 1, 2, '2022-01-01', 3, 10.47),
        (2, 2, 5, '2022-01-02', 2, 9.98),
        (3, 3, 7, '2022-01-03', 1, 8.99),
        (4, 4, 4, '2022-01-04', 4, 11.96),
        (5, 5, 1, '2022-01-05', 2, 11.98),
        (6, 6, 6, '2022-01-06', 3, 13.47),
        (7, 7, 9, '2022-01-07', 1, 6.99),
        (8, 8, 8, '2022-01-08', 2, 6.98),
        (9, 9, 3, '2022-01-09', 1, 10.99),
        (10, 10, 10, '2022-01-10', 1, 7.99);


INSERT INTO Payments (payment_id, order_id, payment_date, payment_amount,
payment_method) VALUES
        (1, 1, '2022-01-02', 10.47, 'Credit Card'),
        (2, 2, '2022-01-03', 9.98, 'PayPal'),
        (3, 3, '2022-01-04', 8.99, 'Venmo'),
        (4, 4, '2022-01-05', 11.96, 'Cash'),
        (5, 5, '2022-01-06', 11.98, 'Credit Card'),
        (6, 6, '2022-01-07', 13.47, 'PayPal'),
        (7, 7, '2022-01-08', 6.99, 'Venmo'),
        (8, 8, '2022-01-09', 6.98, 'Cash'),
        (9, 9, '2022-01-10', 10.99, 'Credit Card'),
        (10, 10, '2022-01-11', 7.99, 'PayPal');
```

```
INSERT INTO Invoices (invoice_id, order_id,invoice_date, due_date, total_amount,
discount_amount, tax_amount, paid_amount, payment_status)
VALUES
    (1, 1, '2022-01-02', '2022-01-16', 10.47, 0, 0.9443, 10.47, 'Paid'),
    (2, 2, '2022-01-03', '2022-01-17', 9.98, 0, 0.8984, 9.98, 'Paid'),
    (3, 3,'2022-01-04', '2022-01-18', 8.99, 0, 0.8091, 8.99, 'Paid'),
    (4, 4, '2022-01-05', '2022-01-19', 11.96, 0, 1.0764, 11.96, 'Paid'),
    (5, 5, '2022-01-06', '2022-01-20', 11.98, 0, 1.0782, 11.98, 'Paid'),
    (6, 6,  '2022-01-07', '2022-01-21', 13.47, 0, 1.2123, 13.47, 'Paid'),
    (7, 7, '2022-01-08', '2022-01-22', 6.99, 0, 0.6291, 6.99, 'Paid'),
    (8, 8,  '2022-01-09', '2022-01-23', 6.98, 0, 0.6282, 6.98, 'Paid'),
    (9, 9,  '2022-01-10', '2022-01-24', 10.99, 0, 0.9891, 10.99, 'UnPaid'),
    (10, 10, '2022-01-11', '2022-01-25', 7.99, 0, 0.7191, 7.99, 'Paid');

INSERT INTO Invoice_Details (
    "invoice_id","payment_id",”discount_id")
VALUES
    (1,1,11),
    ( 2, 2,11),
    ( 3,3,11),
    ( 4,4,11),
    ( 5,5,11),
    (6, 6,11),
    ( 7, 7,11),
    (8,  8,11),
    (9, 9,11),
    (10, 10,11);
    );

INSERT INTO Sales (sale_id, customer_id, employee_id, date_time, total_amount)
VALUES
    (1, 1, 2, '2022-01-01 10:00:00', 50.0),
    (2, 3, 4, '2022-01-02 11:30:00', 75.0),
    (3, 2, 3, '2022-01-03 12:45:00', 30.0),
    (4, 5, 1, '2022-01-04 14:20:00', 20.0),
    (5, 4, 2, '2022-01-05 15:10:00', 45.0),
    (6, 6, 1, '2022-01-06 16:30:00', 60.0),
    (7, 8, 4, '2022-01-07 17:15:00', 55.0),
    (8, 7, 3, '2022-01-08 18:20:00', 40.0),
    (9, 10, 2, '2022-01-09 19:40:00', 25.0),
    (10, 9, 1, '2022-01-10 20:50:00', 35.0);
```

The SQL statements corresponding to each of your requirements. (If your requirements call for more than a dozen or so statements, you can do a subset). Your SQL statements should require a variety of SQL capabilities, such as various kinds of join, aggregate functions, etc. (This presupposes a good initial domain choice.)
http://neelkumarpatel.com/Projects/Advance_database/Mid_project/Select_Statement.sql

1)Select all data from Customers table

```
SELECT * FROM Customers;
```

2)Select data from Customers table where customer_id = 3

```
SELECT * FROM Customers WHERE customer_id = 3;
```

3)Select data from Products table where price is greater than 5

```
SELECT * FROM Products WHERE price > 5;
```

4)Select data from Products table where category_id is either 1 or 2

```
SELECT * FROM Products WHERE category_id IN (1, 2);
```

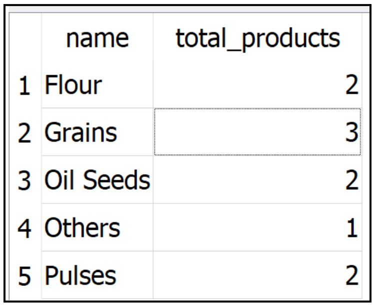
5)Select data from Products table ordered by price in descending order

```
SELECT * FROM Products ORDER BY price DESC;
```

6)Select the total number of products in each category

```
SELECT Category.name, COUNT(Products.product_id) AS total_products
FROM Products
INNER JOIN Category ON Products.category_id = Category.category_id
GROUP BY Category.name;
```

Output:

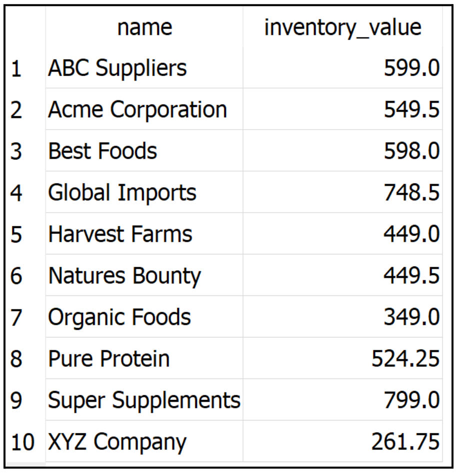


	name	total_products
1	Flour	2
2	Grains	3
3	Oil Seeds	2
4	Others	1
5	Pulses	2

7)Select the total inventory value for each supplier

```
SELECT Suppliers.name, SUM(Products.inventory_level * Products.price) AS inventory_value
FROM Products
INNER JOIN Suppliers ON Products.supplier_id = Suppliers.supplier_id
GROUP BY Suppliers.name;
```

Output:



	name	inventory_value
1	ABC Suppliers	599.0
2	Acme Corporation	549.5
3	Best Foods	598.0
4	Global Imports	748.5
5	Harvest Farms	449.0
6	Natures Bounty	449.5
7	Organic Foods	349.0
8	Pure Protein	524.25
9	Super Supplements	799.0
10	XYZ Company	261.75

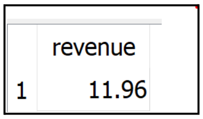
8) Retrieve all products with a weight greater than 10 lbs:

```
SELECT * FROM Products WHERE weight > 10;
```

9)Retrieve the total revenue generated by a specific product:

```
SELECT SUM(total_amount) AS revenue
FROM Orders
WHERE product_id ='4';
```

Output:

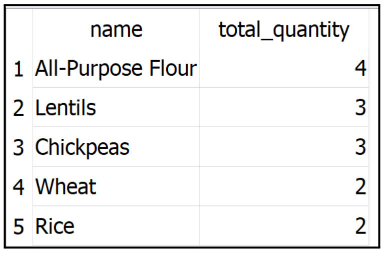


	revenue
1	11.96

10)Retrieve the top 5 best-selling products:

```
SELECT Products.name, SUM(Orders.quantity) AS total_quantity
FROM Products
JOIN Orders ON Products.product_id = Orders.product_id
GROUP BY Products.name
ORDER BY total_quantity DESC
LIMIT 5;
```

Output:

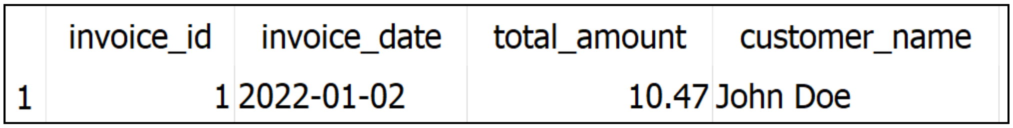


	name	total_quantity
1	All-Purpose Flour	4
2	Lentils	3
3	Chickpeas	3
4	Wheat	2
5	Rice	2

11)Retrieve all invoices for a specific customer:

```
SELECT Invoices.invoice_id, Invoices.invoice_date, Invoices.total_amount , Customers.name
AS customer_name
FROM Invoices
JOIN Orders ON Invoices.order_id = Orders.order_id
JOIN Customers ON Orders.customer_id = Customers.customer_id
WHERE Customers.customer_id = 1;
```

Output:

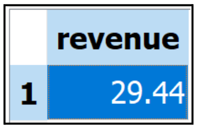


	invoice_id	invoice_date	total_amount	customer_name
1	1	2022-01-02	10.47	John Doe

12) Retrieve the total revenue generated by all sales in a given time period:

```
SELECT SUM(total_amount) AS revenue
FROM Invoices
WHERE invoice_date BETWEEN '2022-01-02' AND '2022-01-04';
```

Output:

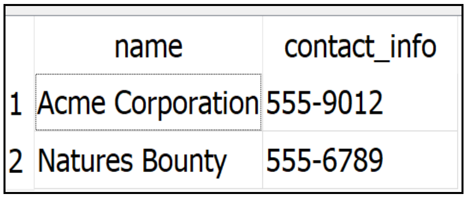


	revenue
1	29.44

13) Retrieve the name and contact information of all suppliers who provide products in a specific category:

```
SELECT DISTINCT Suppliers.name, Suppliers.contact_info
FROM Suppliers
JOIN Products ON Products.supplier_id = Suppliers.supplier_id
WHERE Products.category_id = ‘3’;
```

Output:

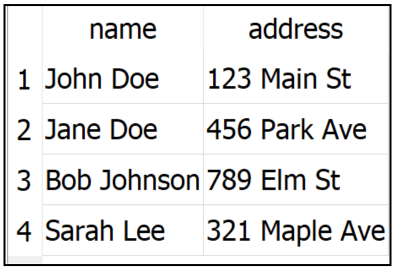


	name	contact_info
1	Acme Corporation	555-9012
2	Natures Bounty	555-6789

14) Retrieve the name and address of all customers who have placed an order in a specific time period:

```
SELECT DISTINCT Customers.name, Customers.address
FROM Customers
JOIN Orders ON Orders.customer_id = Customers.customer_id
WHERE Orders.order_date BETWEEN [start_date] AND [end_date];
```

Output:



	name	address
1	John Doe	123 Main St
2	Jane Doe	456 Park Ave
3	Bob Johnson	789 Elm St
4	Sarah Lee	321 Maple Ave

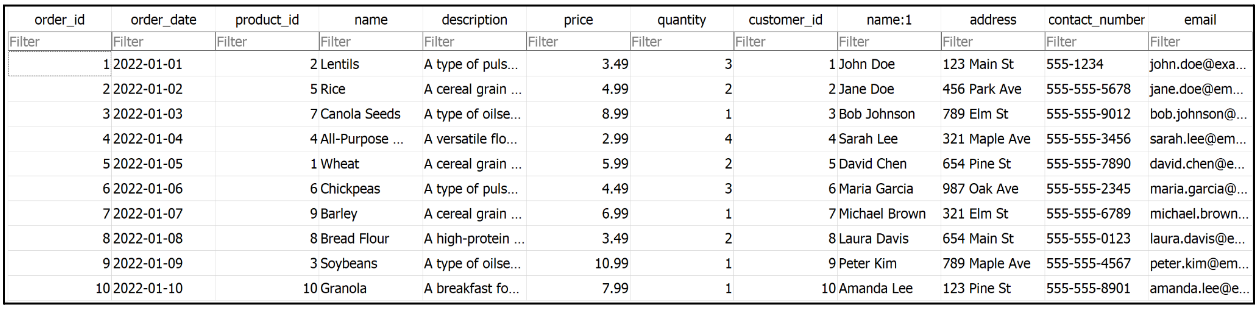
Views

http://neelkumarpatel.com/Projects/Advance_database/Mid_project/views.sql

1)View to show customer order history

```
CREATE VIEW customer_order_history AS
SELECT
    o.order_id,o.order_date,o.product_id,p.name,p.description,p.price,o.quantity,
    c.customer_id,c.name,c.address,c.contact_number,c.email
FROM
    orders o
    JOIN customers c
    ON o.customer_id = c.customer_id
    JOIN products p
    ON o.product_id = p.product_id;
```

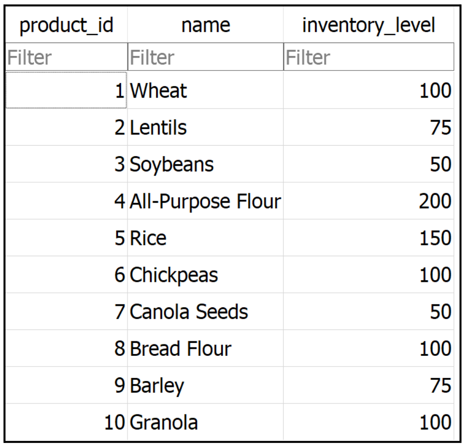
Output:



order_id	order_date	product_id	name	description	price	quantity	customer_id	name:1	address	contact_number	email
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	2022-01-01	2	Lentils	A type of puls...	3.49	3	1	John Doe	123 Main St	555-1234	john.doe@exa...
2	2022-01-02	5	Rice	A cereal grain ...	4.99	2	2	Jane Doe	456 Park Ave	555-555-5678	jane.doe@em...
3	2022-01-03	7	Canola Seeds	A type of oilse...	8.99	1	3	Bob Johnson	789 Elm St	555-555-9012	bob.johnson@...
4	2022-01-04	4	All-Purpose ...	A versatile flo...	2.99	4	4	Sarah Lee	321 Maple Ave	555-555-3456	sarah.lee@em...
5	2022-01-05	1	Wheat	A cereal grain ...	5.99	2	5	David Chen	654 Pine St	555-555-7890	david.chen@e...
6	2022-01-06	6	Chickpeas	A type of puls...	4.49	3	6	Maria Garcia	987 Oak Ave	555-555-2345	maria.garcia@...
7	2022-01-07	9	Barley	A cereal grain ...	6.99	1	7	Michael Brown	321 Elm St	555-555-6789	michael.brown...
8	2022-01-08	8	Bread Flour	A high-protein ...	3.49	2	8	Laura Davis	654 Main St	555-555-0123	laura.davis@e...
9	2022-01-09	3	Soybeans	A type of oilse...	10.99	1	9	Peter Kim	789 Maple Ave	555-555-4567	peter.kim@em...
10	2022-01-10	10	Granola	A breakfast fo...	7.99	1	10	Amanda Lee	123 Pine St	555-555-8901	amanda.lee@e...

2)View to show product inventory levels

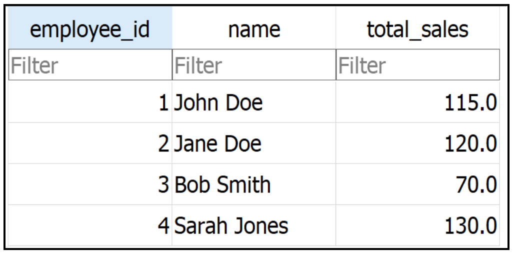
CREATE VIEW **product_inventory_levels** AS
SELECT p.product_id, p.name, p.inventory_level
FROM Products p;
Output:



product_id	name	inventory_level
1	Wheat	100
2	Lentils	75
3	Soybeans	50
4	All-Purpose Flour	200
5	Rice	150
6	Chickpeas	100
7	Canola Seeds	50
8	Bread Flour	100
9	Barley	75
10	Granola	100

3)View to show total sales by employee:

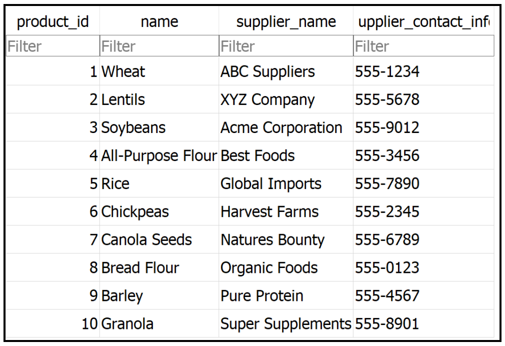
CREATE VIEW **total_sales_by_employee** AS
SELECT e.employee_id, e.name, SUM(s.total_amount) AS total_sales
FROM Sales s
JOIN Employees e ON s.employee_id = e.employee_id
GROUP BY e.employee_id;
Output:



employee_id	name	total_sales
1	John Doe	115.0
2	Jane Doe	120.0
3	Bob Smith	70.0
4	Sarah Jones	130.0

4)View to show supplier information for a product

CREATE VIEW **product_supplier_info** AS
SELECT p.product_id, p.name, s.name AS supplier_name, s.contact_info AS supplier_contact_info
FROM Products p
JOIN Suppliers s ON p.supplier_id = s.supplier_id;
Output:

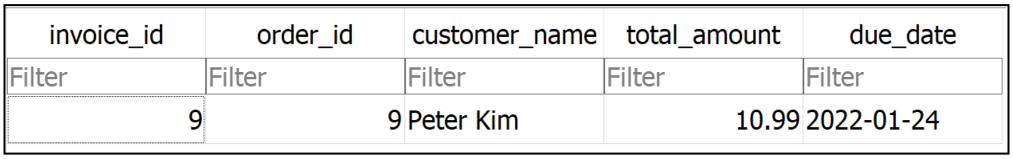


product_id	name	supplier_name	upplier_contact_inf
Filter	Filter	Filter	Filter
1	Wheat	ABC Suppliers	555-1234
2	Lentils	XYZ Company	555-5678
3	Soybeans	Acme Corporation	555-9012
4	All-Purpose Flour	Best Foods	555-3456
5	Rice	Global Imports	555-7890
6	Chickpeas	Harvest Farms	555-2345
7	Canola Seeds	Natures Bounty	555-6789
8	Bread Flour	Organic Foods	555-0123
9	Barley	Pure Protein	555-4567
10	Granola	Super Supplements	555-8901

5)View to show unpaid invoices

```
CREATE VIEW unpaid_invoices AS
SELECT i.invoice_id, o.order_id, c.name AS customer_name, i.total_amount, i.due_date
FROM Invoices i
JOIN Orders o ON i.order_id = o.order_id
JOIN Customers c ON o.customer_id = c.customer_id
WHERE i.paid_amount < i.total_amount;
```

Output:

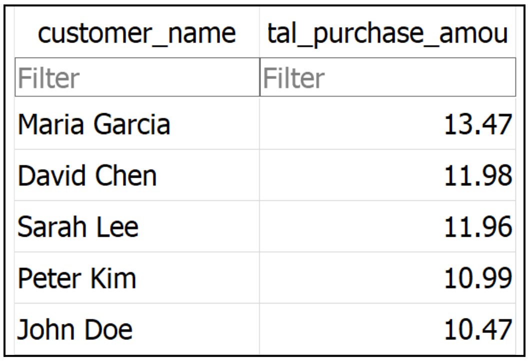


invoice_id	order_id	customer_name	total_amount	due_date
Filter	Filter	Filter	Filter	Filter
9	9	Peter Kim	10.99	2022-01-24

6)This view can show the top 5 customers based on their total purchase amount.

```
CREATE VIEW top_5_customers AS
SELECT c.name AS customer_name, SUM(o.total_amount) AS total_purchase_amount
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id
ORDER BY total_purchase_amount DESC LIMIT 5;
```

Output:

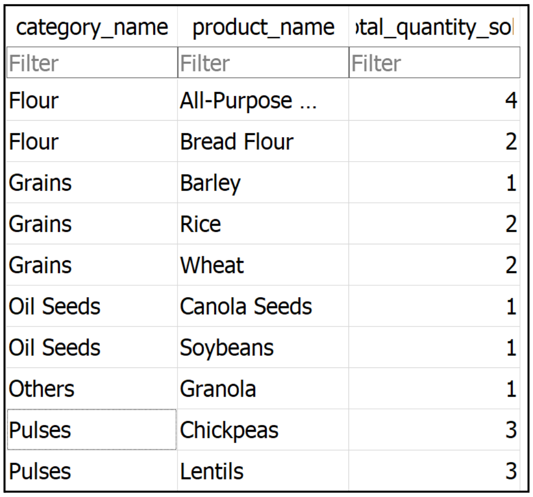


customer_name	tal_purchase_amou
Filter	Filter
Maria Garcia	13.47
David Chen	11.98
Sarah Lee	11.96
Peter Kim	10.99
John Doe	10.47

7)View to show product sales by category

```
CREATE VIEW product_sales_by_category AS
SELECT c.name AS category_name, p.name AS product_name, SUM(o.quantity) AS
total_quantity_sold
FROM Products p
JOIN Category c ON p.category_id = c.category_id
JOIN Orders o ON p.product_id = o.product_id
GROUP BY c.name, p.name;
```

Output:



category_name	product_name	tal_quantity_sol
Filter	Filter	Filter
Flour	All-Purpose ...	4
Flour	Bread Flour	2
Grains	Barley	1
Grains	Rice	2
Grains	Wheat	2
Oil Seeds	Canola Seeds	1
Oil Seeds	Soybeans	1
Others	Granola	1
Pulses	Chickpeas	3
Pulses	Lentils	3

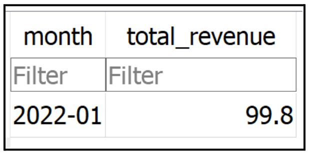
8)This view can show the list of products that are out of stock.

```
CREATE VIEW out_of_stock_products AS
SELECT p.name AS product_name, p.inventory_level AS current_inventory
FROM Products p
WHERE p.inventory_level = 0;
```

9)Total Revenue by Month

```
--This view can show the total revenue for each month.
CREATE VIEW revenue_by_month AS
SELECT strftime('%Y-%m', o.order_date) AS month, SUM(o.total_amount) AS total_revenue
FROM Orders o
GROUP BY strftime('%Y-%m', o.order_date)
ORDER BY month;
```

Output:

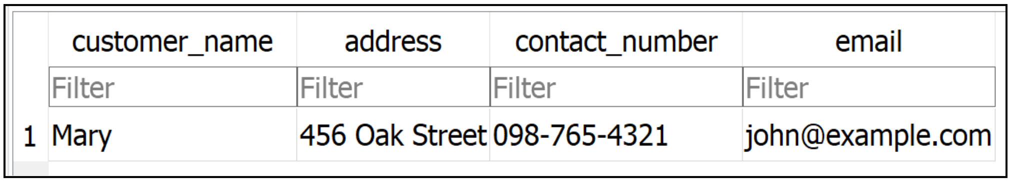


month	total_revenue
Filter	Filter
2022-01	99.8

10)This view can show the list of customers who haven't placed any orders.

```
CREATE VIEW customers_without_orders AS
SELECT c.name AS customer_name, c.address, c.contact_number, c.email
FROM Customers c
LEFT JOIN Orders o ON c.customer_id = o.customer_id
WHERE o.order_id IS NULL;
```

Output:



	customer_name	address	contact_number	email
	Filter	Filter	Filter	Filter
1	Mary	456 Oak Street	098-765-4321	john@example.com

Triggers

http://neelkumarpatel.com/Projects/Advance_database/Mid_project/triggers.sq

1)Trigger to update inventory of products:

```
CREATE TRIGGER update_inventory_level
AFTER INSERT ON Orders
FOR EACH ROW
BEGIN
  UPDATE Products SET inventory_level = inventory_level - NEW.quantity
  WHERE product_id = NEW.product_id;
END;
```

2)Trigger to update total amount in Invoices table after an order is placed:

```
CREATE TRIGGER update_total_amount
AFTER INSERT ON Orders
BEGIN
  INSERT INTO Invoices (order_id, payment_id, discount_id, customer_id, invoice_date,
due_date, total_amount, discount_amount, tax_amount, paid_amount, payment_status)
  VALUES (NEW.order_id, NULL, NULL, NEW.customer_id, NEW.order_date, NULL,
NEW.quantity * Products.price, 0, 0, 0, 'unpaid');
END;
```

Testing and Validation of SQL Statements

1) Customers Table:

--Inserting valid data:

```
INSERT INTO Customers (customer_id, name, address, contact_number, email) VALUES (1,
'John', '123 Main Street', '123-456-7890', 'john@example.com');
```

--Inserting duplicate email (violating unique constraint):

```
INSERT INTO Customers (customer_id, name, address, contact_number, email) VALUES (12,
'Mary', '456 Oak Street', '098-765-4321', 'john@example.com');
```

--Result: Error message "UNIQUE constraint failed: Customers.email"

2) Products Table:

--Inserting valid data:

```
INSERT INTO Products (product_id, category_id, name, description, price, weight, ups,
inventory_level, supplier_id) VALUES (1, 1, 'Product 1', 'Description 1', 10.99, 0.5, '123456', 10,
1);
```

Inserting invalid weight (violating check constraint):

```
INSERT INTO Products (product_id, category_id, name, description, price, weight, ups,
inventory_level, supplier_id) VALUES (2, 2, 'Product 2', 'Description 2', 20.99, -1, '654321', 5,
2);
```

--Result: Error message "CHECK constraint failed: Products.weight"

3) Orders Table:

--Inserting valid data:

```
INSERT INTO Orders (order_id, customer_id, product_id, order_date, quantity, total_amount)
VALUES (1, 1, 1, '2023-03-07', 2, 21.98);
```

–Inserting negative quantity (violating check constraint):

```
INSERT INTO Orders (order_id, customer_id, product_id, order_date, quantity, total_amount)
VALUES (2, 2, 2, '2023-03-07', -1, 0);
```

--Result: Error message "CHECK constraint failed: Orders.quantity"

4)Invoices Table:

--Inserting valid data:
INSERT INTO Invoices (invoice_id, order_id, invoice_date, due_date, total_amount, discount_amount, tax_amount, paid_amount, payment_status) VALUES (1, 1, '2023-03-07', '2023-03-14', 20.00, 2.00, 1.00, 17.00, 'Paid');

--Inserting negative total amount (violating check constraint):
INSERT INTO Invoices (invoice_id, order_id, invoice_date, due_date, total_amount, discount_amount, tax_amount, paid_amount, payment_status) VALUES (2, 2, '2023-03-07', '2023-03-14', -10.00, 0, 0, 0, 'Unpaid');
--Result: Error message "CHECK constraint failed: Invoices.total_amount"

5) Invoice_Details Table:

--Inserting valid data:
INSERT INTO Invoice_Details (invoice_id, payment_id, discount_id) VALUES (1, 1, 1);
--Inserting duplicate invoice id (violating primary key constraint):

INSERT INTO Invoice_Details (invoice_id, payment_id, discount_id) VALUES (1, 2, 2);
--Result: Error message "UNIQUE constraint failed: Invoice_Details.invoice_id"

6)Payment Table:

-- Test that the "payment_amount" column cannot be negative
INSERT INTO Payments(payment_id, order_id, payment_date, payment_amount, payment_method) VALUES(4, 1, '2022-02-01', -100.0, 'Credit Card');

Result: This should fail with a "CHECK constraint failed" error.

-- Test that a payment with a non-existing "order_id" will cause a foreign key constraint error
INSERT INTO Payments(payment_id, order_id, payment_date, payment_amount, payment_method) VALUES(6, 1000, '2022-02-01', 100.0, 'Credit Card');

Result: This should fail with a "FOREIGN KEY constraint failed" error.

-- Test that attempting to insert a payment with a negative payment amount will cause a check constraint error
INSERT INTO Payments(payment_id, order_id, payment_date, payment_amount, payment_method) VALUES(7, 1, '2022-02-01', -100.0, 'Credit Card');

Result: This should fail with a "CHECK constraint failed" error.

7) Employees Table:

-- Test that the "user_name" column must be unique

```
INSERT INTO Employees(employee_id, user_name, password, name, job_title, contact_info)
VALUES(2, 'john.doe', 'password', 'John Doe', 'Manager', 'johndoe2@email.com');
```

Result: This should fail with a "UNIQUE constraint failed" error.

Test that a sale with a non-existing "customer_id".

```
INSERT INTO Sales (customer_id, employee_id, date_time, total_amount)
VALUES (999, 2, '2022-01-01 10:00:00', 100.00);
```

Result:will cause a foreign key constraint error

Test that a sale with a non-existing "employee_id"

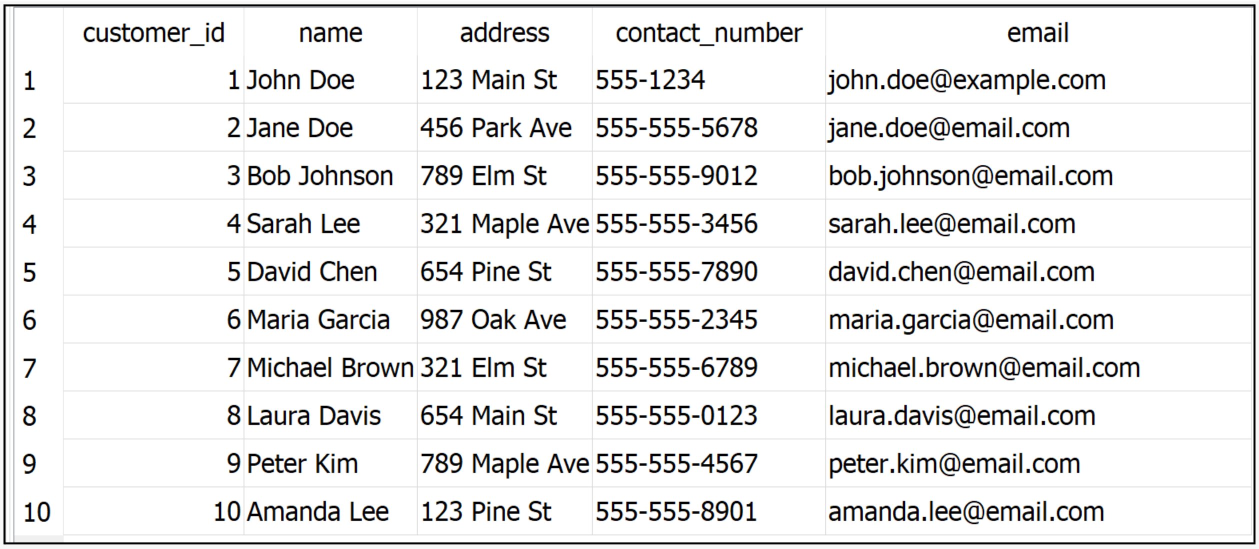
```
INSERT INTO Sales (customer_id, employee_id, date_time, total_amount)
VALUES (1, 999, '2022-01-01 10:00:00', 100.00);
```

Result: will cause a foreign key constraint error

Testing Database

Test Case 1: Retrieve all customer from the Customers table.
SELECT * FROM Customers;

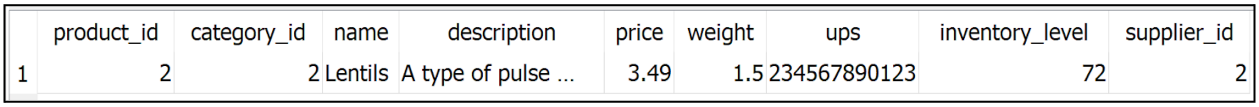
Expected Output: A list of all customers in the Customers Table.



	customer_id	name	address	contact_number	email
1	1	John Doe	123 Main St	555-1234	john.doe@example.com
2	2	Jane Doe	456 Park Ave	555-555-5678	jane.doe@email.com
3	3	Bob Johnson	789 Elm St	555-555-9012	bob.johnson@email.com
4	4	Sarah Lee	321 Maple Ave	555-555-3456	sarah.lee@email.com
5	5	David Chen	654 Pine St	555-555-7890	david.chen@email.com
6	6	Maria Garcia	987 Oak Ave	555-555-2345	maria.garcia@email.com
7	7	Michael Brown	321 Elm St	555-555-6789	michael.brown@email.com
8	8	Laura Davis	654 Main St	555-555-0123	laura.davis@email.com
9	9	Peter Kim	789 Maple Ave	555-555-4567	peter.kim@email.com
10	10	Amanda Lee	123 Pine St	555-555-8901	amanda.lee@email.com

Test Case 2 : Retrieve all products from the products Table for a specific supplier.
SELECT * FROM Products WHERE supplier_id =2;

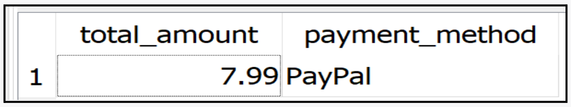
Expected output: A list of all products in the Products table for the specified supplier.



	product_id	category_id	name	description	price	weight	ups	inventory_level	supplier_id
1	2	2	Lentils	A type of pulse ...	3.49	1.5	234567890123	72	2

Test Case 3: Retrieve the total amount and payment method for a specific order from the Orders and Payments tables
SELECT o.total_amount, p.payment_method FROM Orders o INNER JOIN Payments p ON o.order_id = p.order_id WHERE o.order_id = 10

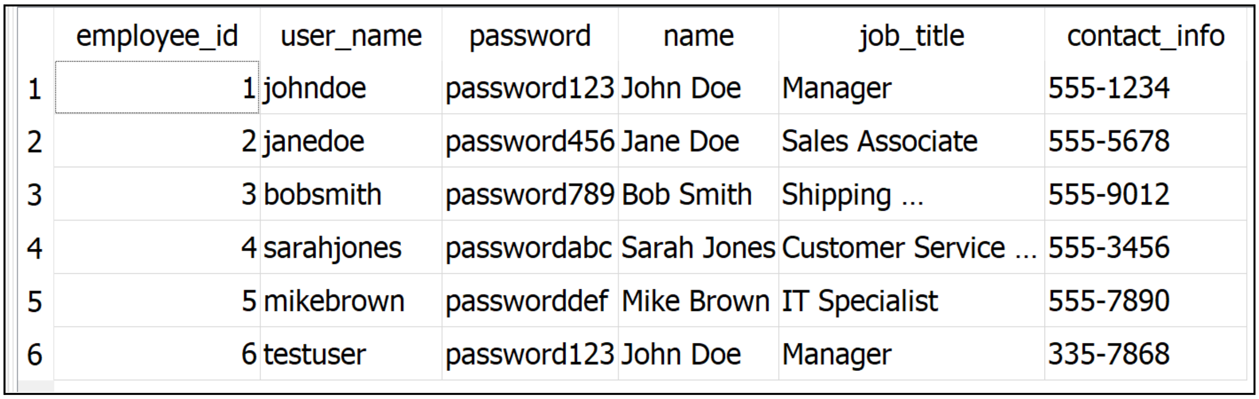
Expected output: The total amount and payment method for the specified order



	total_amount	payment_method
1	7.99	PayPal

Test Case 4: Insert a new employee into the Employees table.
INSERT INTO Employees (user_name, password, name, job_title, contact_info) VALUES ('testuser', 'password123', 'John Doe', 'Manager', 'johndoe@example.com')

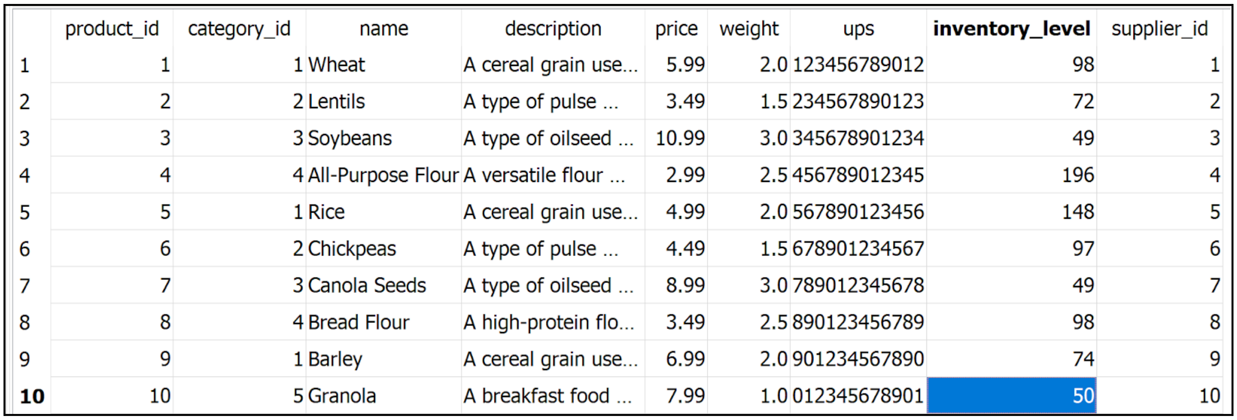
Expected output: A new row should be added to the Employees table with the values specified in the INSERT statement



	employee_id	user_name	password	name	job_title	contact_info
1	1	johndoe	password123	John Doe	Manager	555-1234
2	2	janedoe	password456	Jane Doe	Sales Associate	555-5678
3	3	bobsmith	password789	Bob Smith	Shipping ...	555-9012
4	4	sarahjones	passwordabc	Sarah Jones	Customer Service ...	555-3456
5	5	mikebrown	passworddef	Mike Brown	IT Specialist	555-7890
6	6	testuser	password123	John Doe	Manager	335-7868

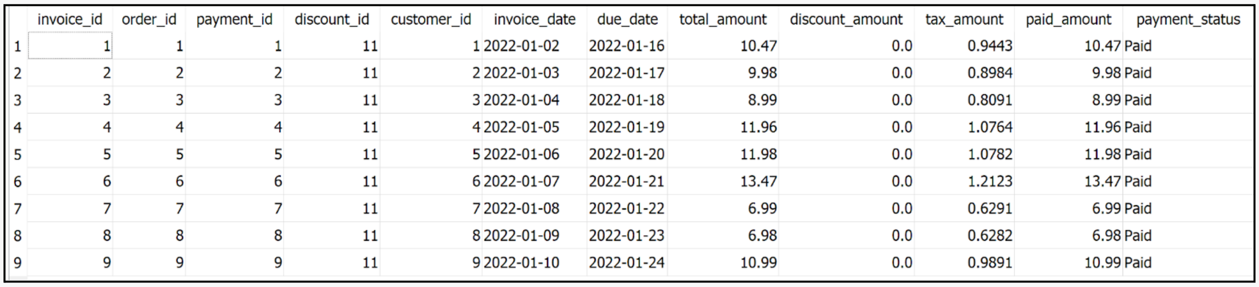
Test Case 5: Update the inventory level for a specific product in the Products table
UPDATE Products SET inventory_level = 50 WHERE product_id = 10;

Expected output: The inventory level for the specified product should be updated to 50.



	product_id	category_id	name	description	price	weight	ups	inventory_level	supplier_id
1	1	1	Wheat	A cereal grain use...	5.99	2.0	123456789012	98	1
2	2	2	Lentils	A type of pulse ...	3.49	1.5	234567890123	72	2
3	3	3	Soybeans	A type of oilseed ...	10.99	3.0	345678901234	49	3
4	4	4	All-Purpose Flour	A versatile flour ...	2.99	2.5	456789012345	196	4
5	5	1	Rice	A cereal grain use...	4.99	2.0	567890123456	148	5
6	6	2	Chickpeas	A type of pulse ...	4.49	1.5	678901234567	97	6
7	7	3	Canola Seeds	A type of oilseed ...	8.99	3.0	789012345678	49	7
8	8	4	Bread Flour	A high-protein flo...	3.49	2.5	890123456789	98	8
9	9	1	Barley	A cereal grain use...	6.99	2.0	901234567890	74	9
10	10	5	Granola	A breakfast food ...	7.99	1.0	012345678901	50	10

Test Case 6: Delete a specific invoice from the Invoices table
DELETE FROM Invoices WHERE invoice_id = 10;
Expected output: The row with the specified invoice_id should be deleted from the Invoices table.



	invoice_id	order_id	payment_id	discount_id	customer_id	invoice_date	due_date	total_amount	discount_amount	tax_amount	paid_amount	payment_status
1	1	1	1	11	1	2022-01-02	2022-01-16	10.47	0.0	0.9443	10.47	Paid
2	2	2	2	11	2	2022-01-03	2022-01-17	9.98	0.0	0.8984	9.98	Paid
3	3	3	3	11	3	2022-01-04	2022-01-18	8.99	0.0	0.8091	8.99	Paid
4	4	4	4	11	4	2022-01-05	2022-01-19	11.96	0.0	1.0764	11.96	Paid
5	5	5	5	11	5	2022-01-06	2022-01-20	11.98	0.0	1.0782	11.98	Paid
6	6	6	6	11	6	2022-01-07	2022-01-21	13.47	0.0	1.2123	13.47	Paid
7	7	7	7	11	7	2022-01-08	2022-01-22	6.99	0.0	0.6291	6.99	Paid
8	8	8	8	11	8	2022-01-09	2022-01-23	6.98	0.0	0.6282	6.98	Paid
9	9	9	9	11	9	2022-01-10	2022-01-24	10.99	0.0	0.9891	10.99	Paid

Conclusion

This report presented the design and implementation of a database for a retail company. The report began by outlining the problem domain and requirements of the database. Next, an Entity-Relationship (ER) diagram and Relational Design diagrams were developed to represent the data model. The database was then converted into the third normal form (3NF) to reduce redundancy and improve data integrity. Several SQL statements, including create, insert, select, views, and trigger statements, were developed to manipulate the database. Finally, the report discussed testing and validation, including constraints and the overall functionality of the database. Overall, the database design and implementation presented in this report meets the requirements of the retail company and provides a robust and efficient solution.