CPSC 332
Final Project Report

Kelly's Bakery

By:

Greg Zhang (CWID: 886431147)

CPSC 332-03 (13894)
Spring 2022

Professor: Keerthi Vardhan Sovenahalli Karanam

Department of Computer Science California State University, Fullerton
April 11, 2022

Table of Contents

ABSTRACT

This database project is for a baker named Kelly who runs a bakery that makes cakes to order for customers. The bakery is named Kelly's Bakery and it requires a database to check on customer's orders and can see which employees are working on what orders. The database also can be used to see the employees and customers' information. It allows the owner to easily keep track of orders and can see deadlines and order status inputted by the employees.

INTRODUCTION

The main goal of this project is solely to create a database for a baker, Kelly, who can use the database to check up on orders placed in her bakery and store employee, customer, and order information. This project will help with the management of her bakery in all aspects. The project is created using MySQL and the database is written entirely in SQL.

Project Goals

1. The owner or manager of the bakery should have a database where all aspects of the bakery can easily be managed.
2. The database should incorporate not only the customer's information but employee's as well.
3. The system must be able to hold past information (receipts) on past orders.
4. Orders can be tracked with information about the product, the customer, the employee assigned to the individual order, important dates regarding the order, and the status of the order as well. The orders table will be the most important table in the entire database.

Project Relevance and Significance

1. Learned how to implement tables using SQL.
2. Gained experience in connecting tables utilizing constraints.
3. Learned how to use DDL commands to manage a database.

This project will consist of 6 tables in which all the goals of the project will be met.

List of Tables in Database

1. **Orders**: orders placed should be stored along with what customer had placed the order, which employee is working on the order, what product has been ordered, status of the order, date of the order, and estimated pickup date for the order.

   orders (order_id, order_status, customer_id, product_id, employee_id, order_total, order_date, pickup_date)

2. **Products list**: a list of the products should store the name of the product and the price of the product.

   products (product_id, product_name, product_cost)

3. **Customer list**: a list of the customers of the bakery. Customer table should hold the customer's first and last name as well as contact information for contact when order is available for pickup.

   customers (customer_id, customer_first_name, customer_last_name, customer_email, customer_phone)

4. **Employee list**: a table in the database should hold a list of the employees. This is a crucial table which holds employee information such as name, social security number, and contact information.

   employees (employee_id, employee_first_name, employee_last_name, employee_phone, employee_SSN)

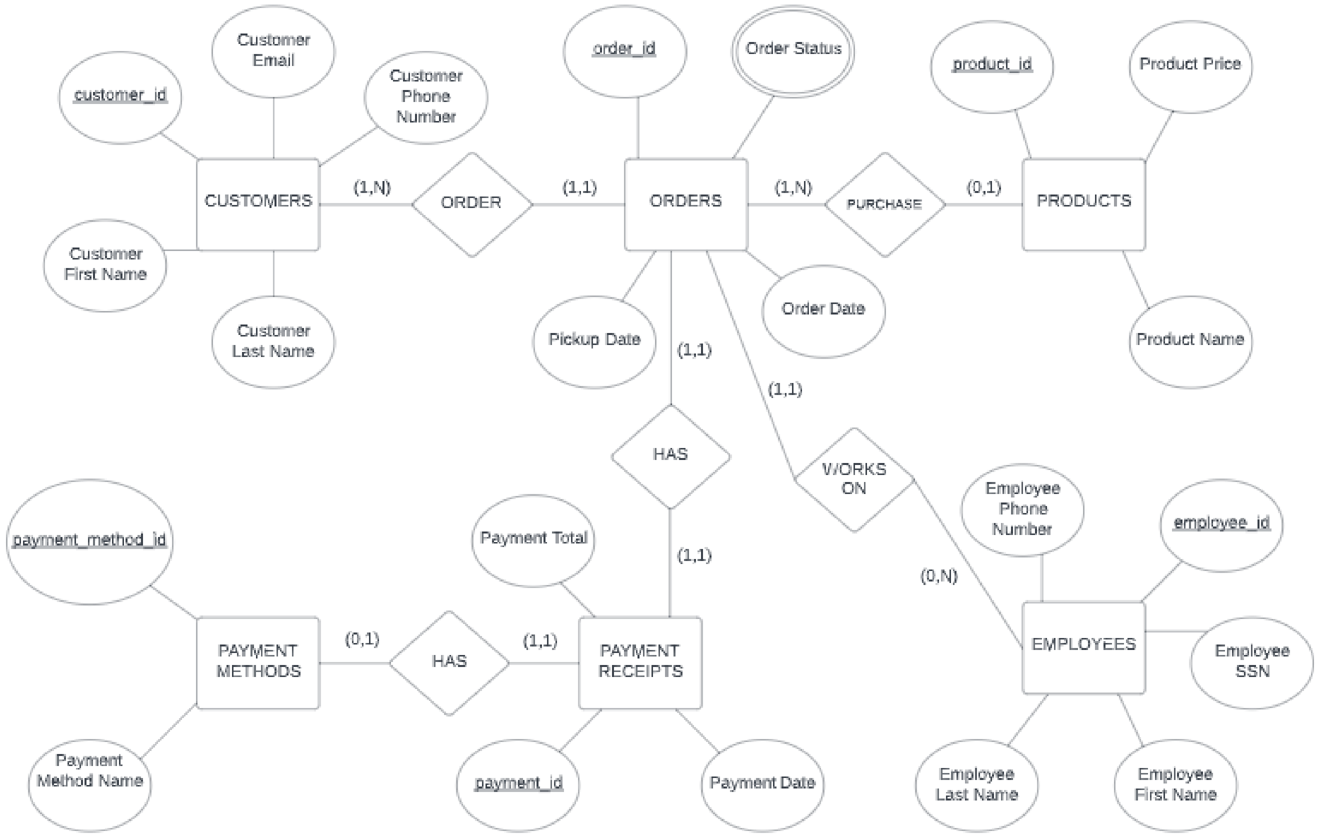
5. **Payment receipts**: a receipt table would hold receipts of all orders where the customer id, quantity, and items will be easily seen; as well as the order total, payment type, and date of payment.

   payments (payment_id, customer_id , order_id, payment_date, payment_total, payment_method)

6. **Payment type list**: a table of available payment methods is necessary for the payment receipts table in such that a user of the database can see what payment methods were used for each order.

   payment_methods (payment_method_id, payment_method_name)

ER DIAGRAM (Using Min, Max Notation)

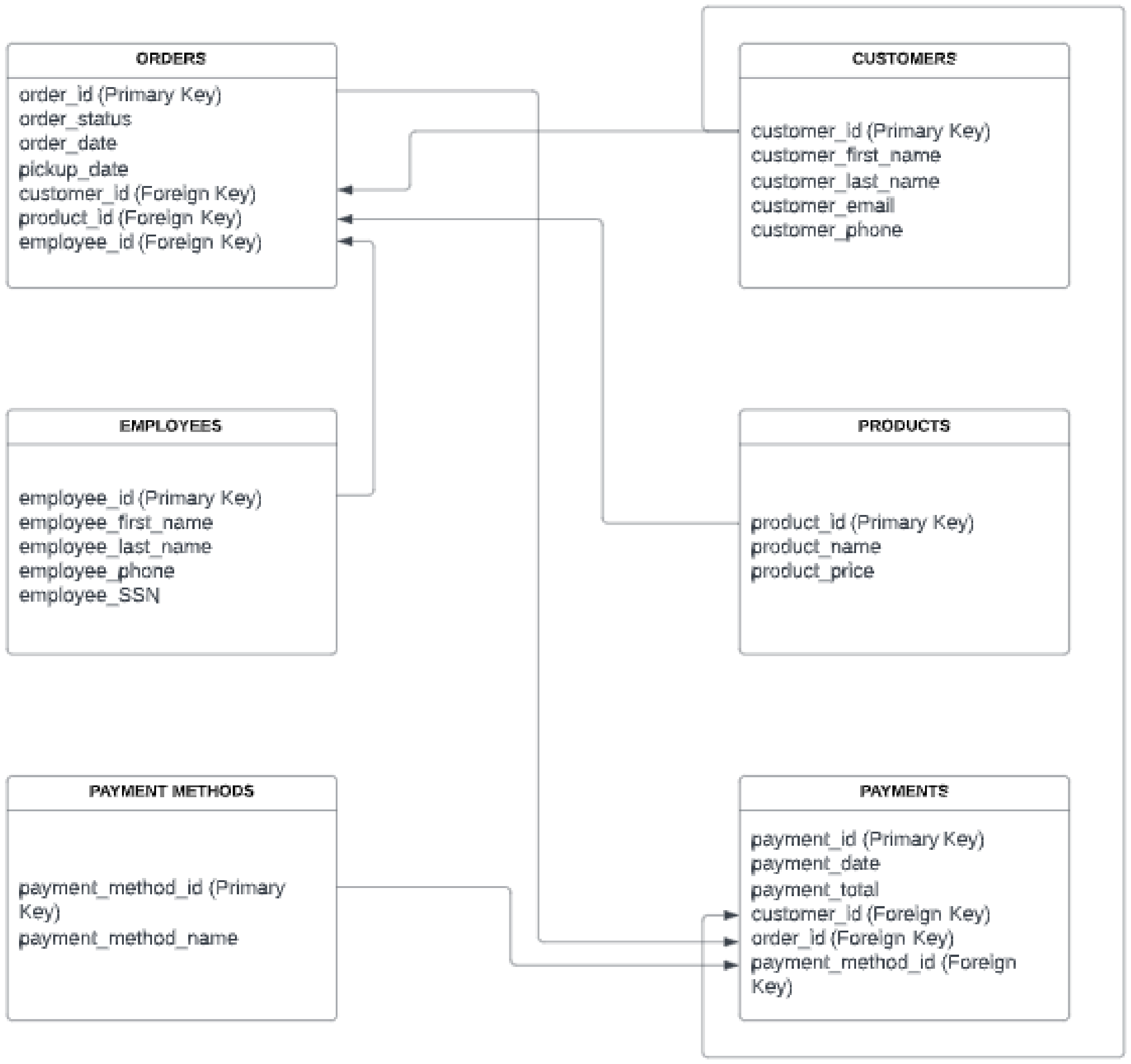


This ER Diagram was created using Lucidchart

Min, Max Notation Explained:

1. Orders can have exactly one customer, but a customer can have as many orders.
   ORDERS: (1,1)   CUSTOMERS: (1,N)
2. Products do not need to have an order, but orders need to have at least one product.
   ORDERS: (1,N)   PRODUCTS: (0,1)
3. Orders need to have a payment receipt, payment receipts need to have one order.
   ORDERS: (1,1)   PAYMENT RECEIPTS: (1,1)
4. Payment methods do not need to have any payment receipts, but payment receipts must have a payment method.
   PAYMENT METHOD: (0,1)   PAYMENT RECEIPTS: (1,1)
5. Orders must have one employee working on the order, but employees do not have to be working on an order or can be working on multiple orders.
   ORDERS: (1,1)   EMPLOYEES: (0,N)

UML CLASS DIAGRAM



This UML Diagram was created using Lucidchart

Refer to the next page for interface requirements.

INTERFACE REQUIREMENTS

Orders

1. order_id: numeric data entry
2. order_status: character data entry
3. order_date: date data entry
4. pickup_date: date data entry

Employees

1. employee_id: numeric data entry
2. employee_first_name: character data entry
3. employee_last_name: character data entry
4. employee_phone: character data entry
5. employee_SSN: character data entry

Payment Methods

1. payment_method_id: numeric data entry
2. payment_method_name: character data entry

Customers

1. customer_id: numeric data entry
2. customer_first_name: character data entry
3. customer_last_name: character data entry
4. customer_email: character data entry
5. customer_phone: character data entry

Products

1. product_id: numeric data entry
2. product_name: character data entry
3. product_price: decimal data entry

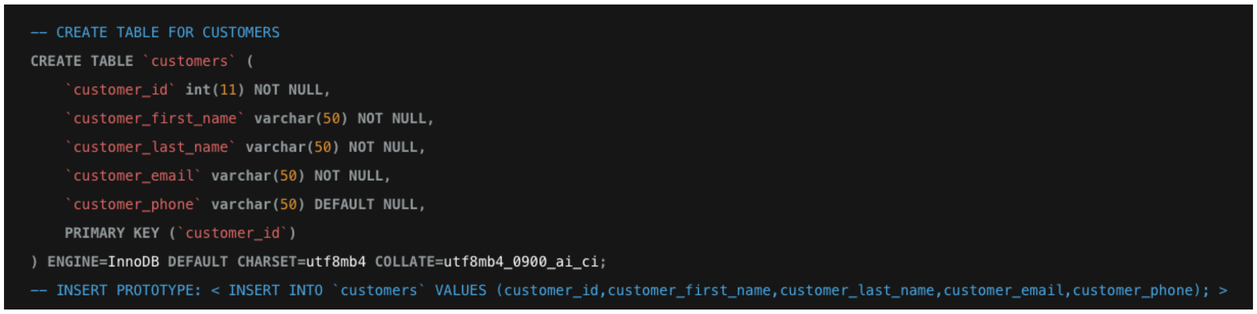
Payments

1. payment_id: numeric data entry
2. payment_date: date data entry
3. payment_total: decimal data entry

METHODOLOGY

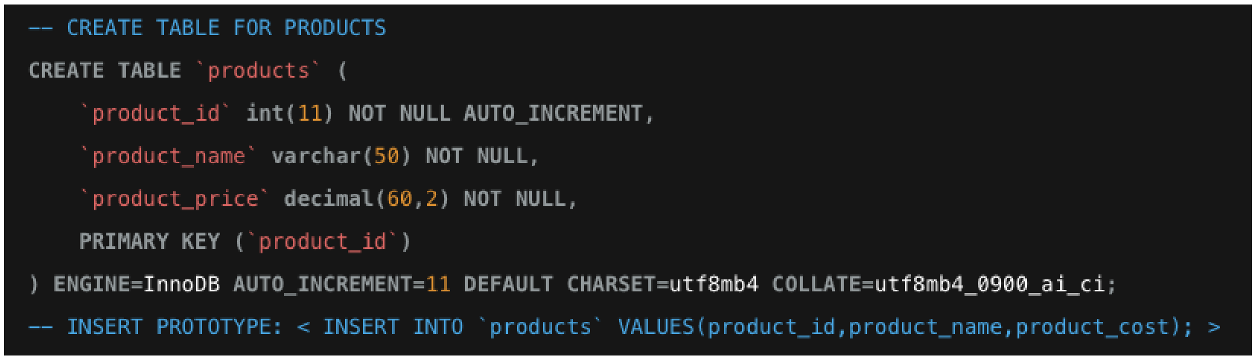
DDL Commands

Customers Table



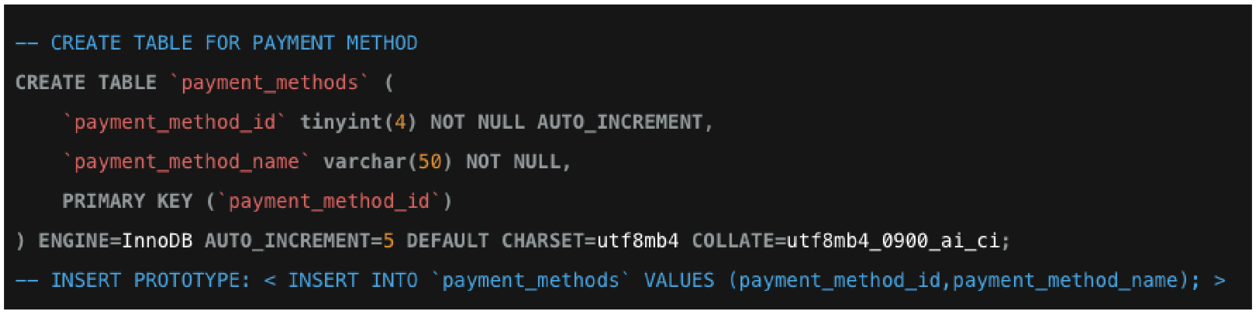
```
-- CREATE TABLE FOR CUSTOMERS
CREATE TABLE `customers` (
    `customer_id` int(11) NOT NULL,
    `customer_first_name` varchar(50) NOT NULL,
    `customer_last_name` varchar(50) NOT NULL,
    `customer_email` varchar(50) NOT NULL,
    `customer_phone` varchar(50) DEFAULT NULL,
    PRIMARY KEY (`customer_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
-- INSERT PROTOTYPE: < INSERT INTO `customers` VALUES (customer_id,customer_first_name,customer_last_name,customer_email,customer_phone); >
```

Products Table



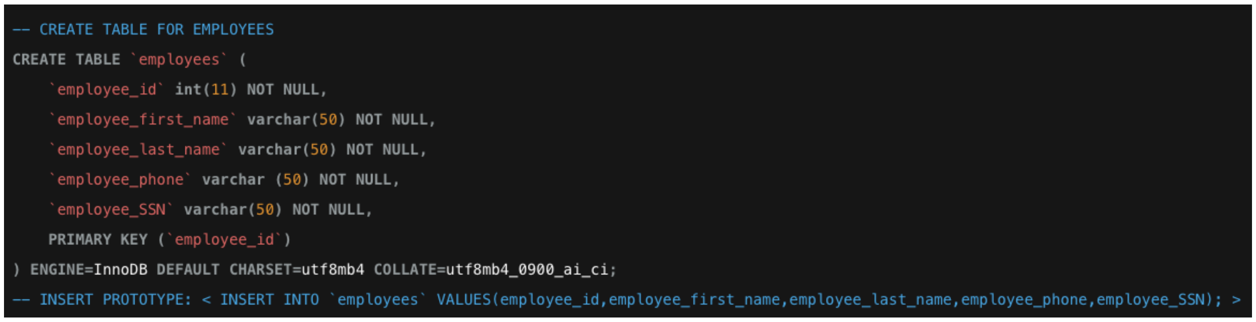
```
-- CREATE TABLE FOR PRODUCTS
CREATE TABLE `products` (
    `product_id` int(11) NOT NULL AUTO_INCREMENT,
    `product_name` varchar(50) NOT NULL,
    `product_price` decimal(60,2) NOT NULL,
    PRIMARY KEY (`product_id`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
-- INSERT PROTOTYPE: < INSERT INTO `products` VALUES(product_id,product_name,product_cost); >
```

Payment Method Table



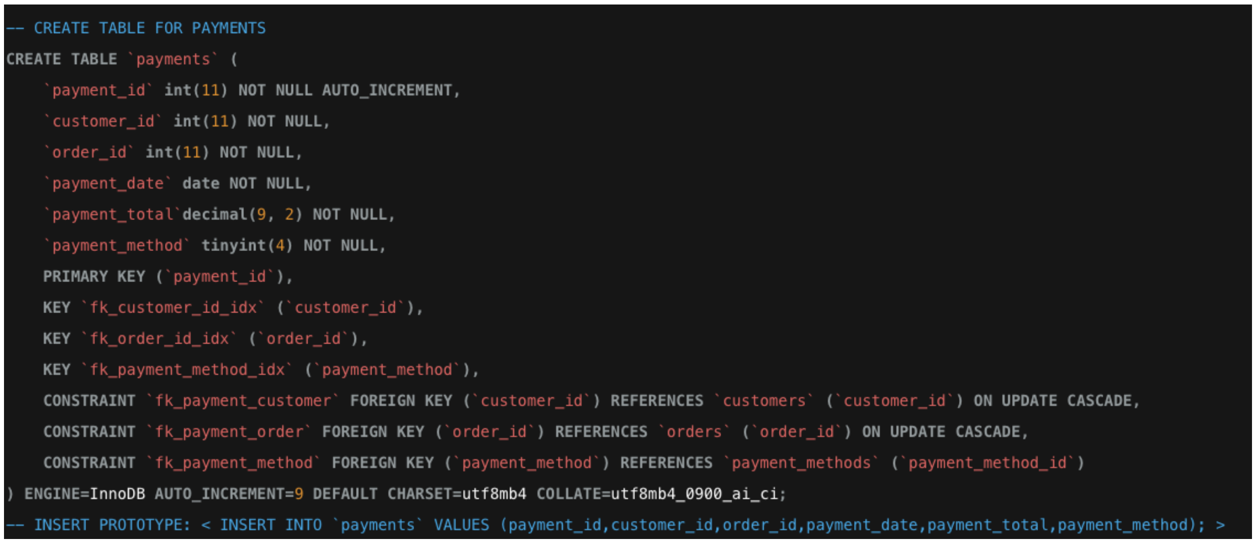
```
-- CREATE TABLE FOR PAYMENT METHOD
CREATE TABLE `payment_methods` (
    `payment_method_id` tinyint(4) NOT NULL AUTO_INCREMENT,
    `payment_method_name` varchar(50) NOT NULL,
    PRIMARY KEY (`payment_method_id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
-- INSERT PROTOTYPE: < INSERT INTO `payment_methods` VALUES (payment_method_id,payment_method_name); >
```

Employees Table



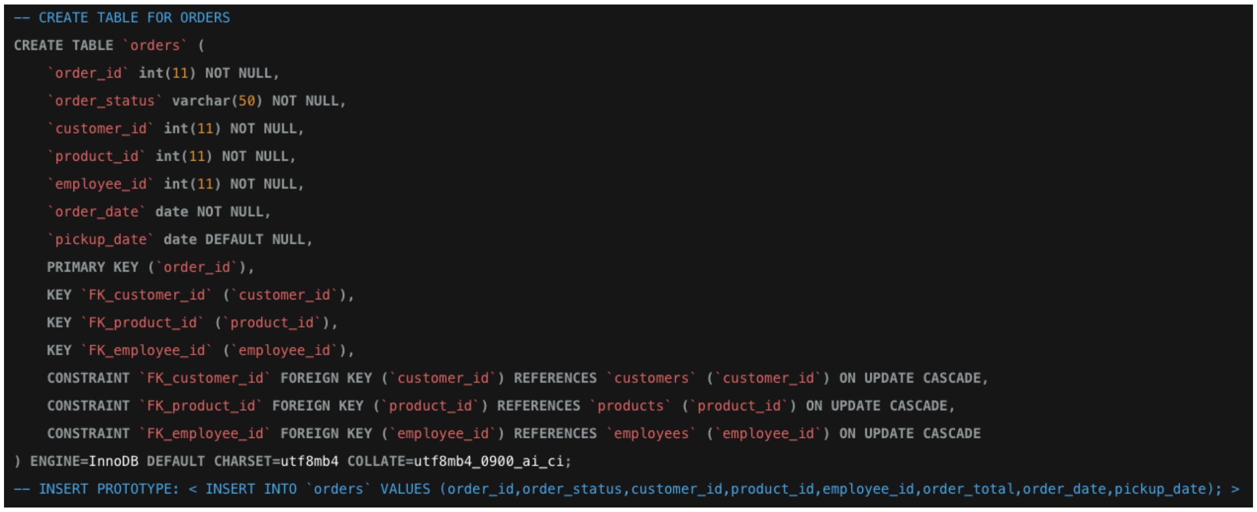
```
-- CREATE TABLE FOR EMPLOYEES
CREATE TABLE `employees` (
    `employee_id` int(11) NOT NULL,
    `employee_first_name` varchar(50) NOT NULL,
    `employee_last_name` varchar(50) NOT NULL,
    `employee_phone` varchar (50) NOT NULL,
    `employee_SSN` varchar(50) NOT NULL,
    PRIMARY KEY (`employee_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
-- INSERT PROTOTYPE: < INSERT INTO `employees` VALUES(employee_id,employee_first_name,employee_last_name,employee_phone,employee_SSN); >
```

Payments Table



```
-- CREATE TABLE FOR PAYMENTS
CREATE TABLE `payments` (
    `payment_id` int(11) NOT NULL AUTO_INCREMENT,
    `customer_id` int(11) NOT NULL,
    `order_id` int(11) NOT NULL,
    `payment_date` date NOT NULL,
    `payment_total`decimal(9, 2) NOT NULL,
    `payment_method` tinyint(4) NOT NULL,
    PRIMARY KEY (`payment_id`),
    KEY `fk_customer_id_idx` (`customer_id`),
    KEY `fk_order_id_idx` (`order_id`),
    KEY `fk_payment_method_idx` (`payment_method`),
    CONSTRAINT `fk_payment_customer` FOREIGN KEY (`customer_id`) REFERENCES `customers` (`customer_id`) ON UPDATE CASCADE,
    CONSTRAINT `fk_payment_order` FOREIGN KEY (`order_id`) REFERENCES `orders` (`order_id`) ON UPDATE CASCADE,
    CONSTRAINT `fk_payment_method` FOREIGN KEY (`payment_method`) REFERENCES `payment_methods` (`payment_method_id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
-- INSERT PROTOTYPE: < INSERT INTO `payments` VALUES (payment_id,customer_id,order_id,payment_date,payment_total,payment_method); >
```

Orders Table



```sql
-- CREATE TABLE FOR ORDERS
CREATE TABLE `orders` (
    `order_id` int(11) NOT NULL,
    `order_status` varchar(50) NOT NULL,
    `customer_id` int(11) NOT NULL,
    `product_id` int(11) NOT NULL,
    `employee_id` int(11) NOT NULL,
    `order_date` date NOT NULL,
    `pickup_date` date DEFAULT NULL,
    PRIMARY KEY (`order_id`),
    KEY `FK_customer_id` (`customer_id`),
    KEY `FK_product_id` (`product_id`),
    KEY `FK_employee_id` (`employee_id`),
    CONSTRAINT `FK_customer_id` FOREIGN KEY (`customer_id`) REFERENCES `customers` (`customer_id`) ON UPDATE CASCADE,
    CONSTRAINT `FK_product_id` FOREIGN KEY (`product_id`) REFERENCES `products` (`product_id`) ON UPDATE CASCADE,
    CONSTRAINT `FK_employee_id` FOREIGN KEY (`employee_id`) REFERENCES `employees` (`employee_id`) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
-- INSERT PROTOTYPE: < INSERT INTO `orders` VALUES (order_id,order_status,customer_id,product_id,employee_id,order_total,order_date,pickup_date); >
```

Software Stack

Database:

MySQL

Software:

MySQLWorkbench

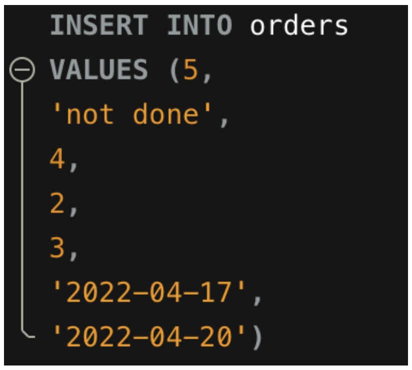
Microsoft Visual Studio Code

Hardware:

Operating System: MacOS Monterey

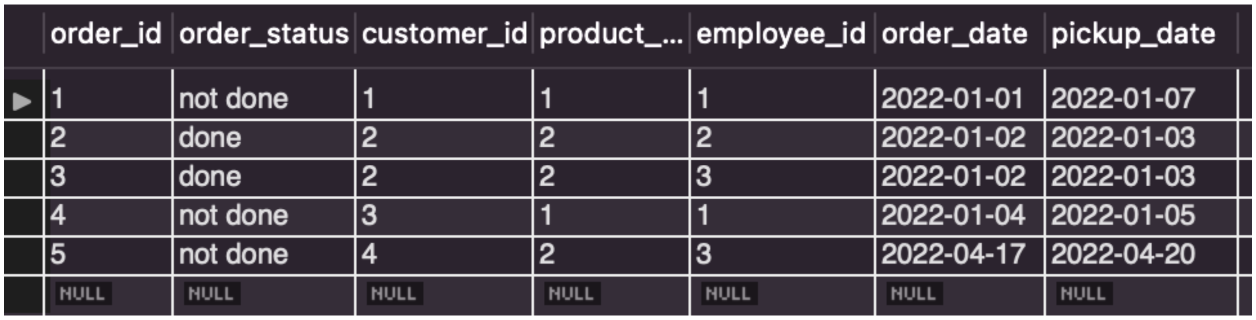
DATABASE BASIC OPERATIONS

Insert Operation



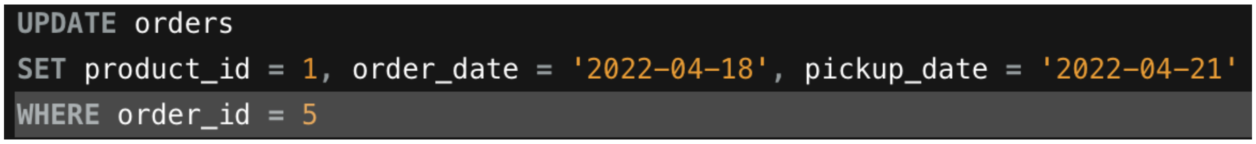
```
INSERT INTO orders
VALUES (5,
'not done',
4,
2,
3,
'2022-04-17',
'2022-04-20')
```

Result: New row was created in the orders table with the inputted values



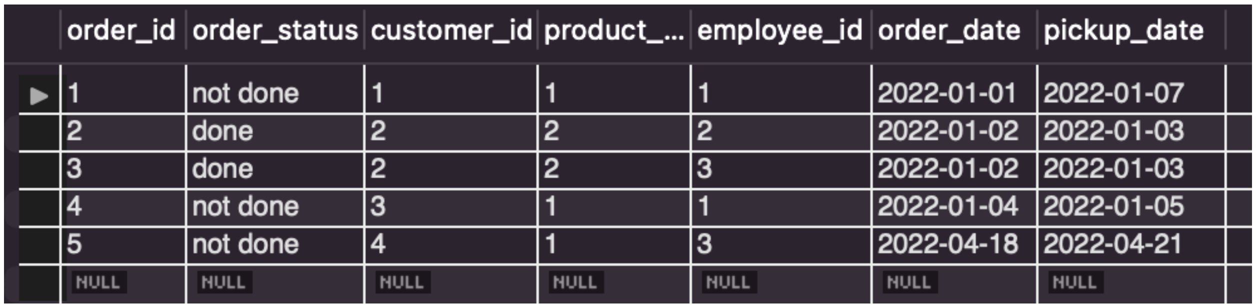
order_id	order_status	customer_id	product_...	employee_id	order_date	pickup_date
1	not done	1	1	1	2022-01-01	2022-01-07
2	done	2	2	2	2022-01-02	2022-01-03
3	done	2	2	3	2022-01-02	2022-01-03
4	not done	3	1	1	2022-01-04	2022-01-05
5	not done	4	2	3	2022-04-17	2022-04-20
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Update Operation



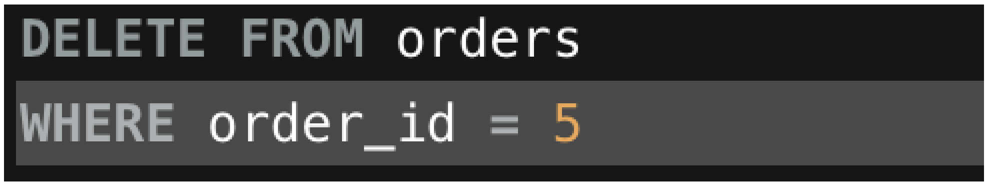
```
UPDATE orders
SET product_id = 1, order_date = '2022-04-18', pickup_date = '2022-04-21'
WHERE order_id = 5
```

Result: Row with order_id = 5 was updated in the columns with the inputted values



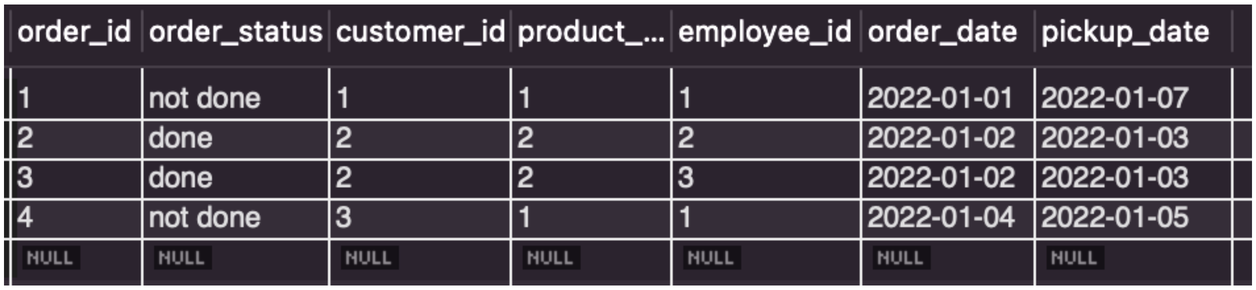
order_id	order_status	customer_id	product_...	employee_id	order_date	pickup_date
1	not done	1	1	1	2022-01-01	2022-01-07
2	done	2	2	2	2022-01-02	2022-01-03
3	done	2	2	3	2022-01-02	2022-01-03
4	not done	3	1	1	2022-01-04	2022-01-05
5	not done	4	1	3	2022-04-18	2022-04-21
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Delete Operation



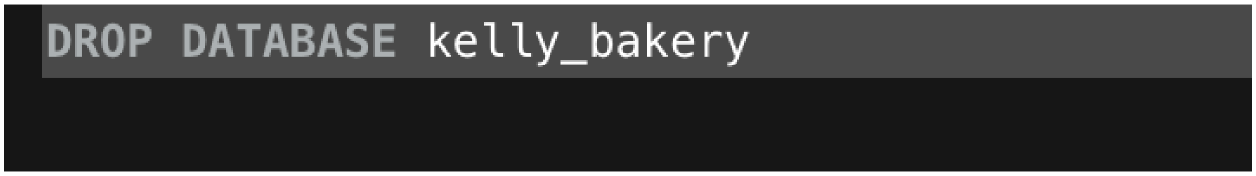
```
DELETE FROM orders
WHERE order_id = 5
```

Result: Row with order_id = 5 was deleted from the orders table



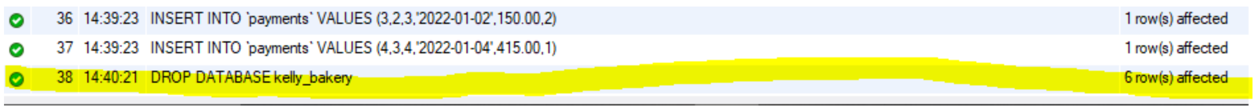
order_id	order_status	customer_id	product_...	employee_id	order_date	pickup_date
1	not done	1	1	1	2022-01-01	2022-01-07
2	done	2	2	2	2022-01-02	2022-01-03
3	done	2	2	3	2022-01-02	2022-01-03
4	not done	3	1	1	2022-01-04	2022-01-05
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Drop Operation (Database)



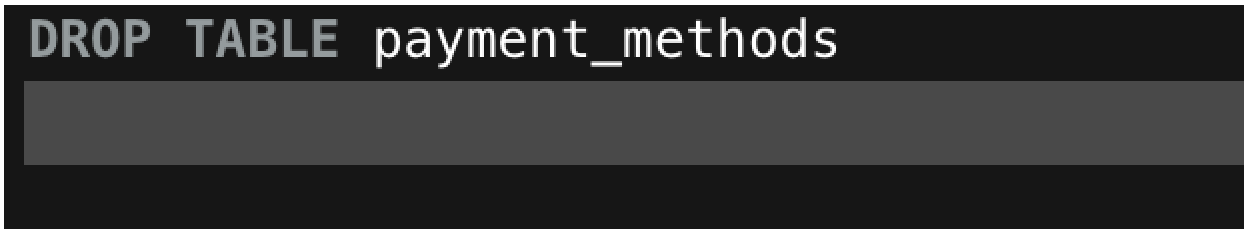
```
DROP DATABASE kelly_bakery
```

Result: Entire database was dropped from the instance



```
36  14:39:23  INSERT INTO `payments` VALUES (3,2,3,'2022-01-02',150.00,2)     1 row(s) affected
37  14:39:23  INSERT INTO `payments` VALUES (4,3,4,'2022-01-04',415.00,1)     1 row(s) affected
38  14:40:21  DROP DATABASE kelly_bakery                                       6 row(s) affected
```

Drop Operation (Table)



```
DROP TABLE payment_methods
```

Result: Payment_methods table was dropped. NOTE: This only works if there are no foreign key constraints in other tables.