

Milestone - 2

Group 10:

Ankur Pandey - +1 (857) 398 5940 /
pandey.anku@northeastern.edu

Giridhar Babu - +1 (857) 398-5730 /
babu.gi@northeastern.edu

Percentage of Effort Contributed by Ankur Pandey: 50%

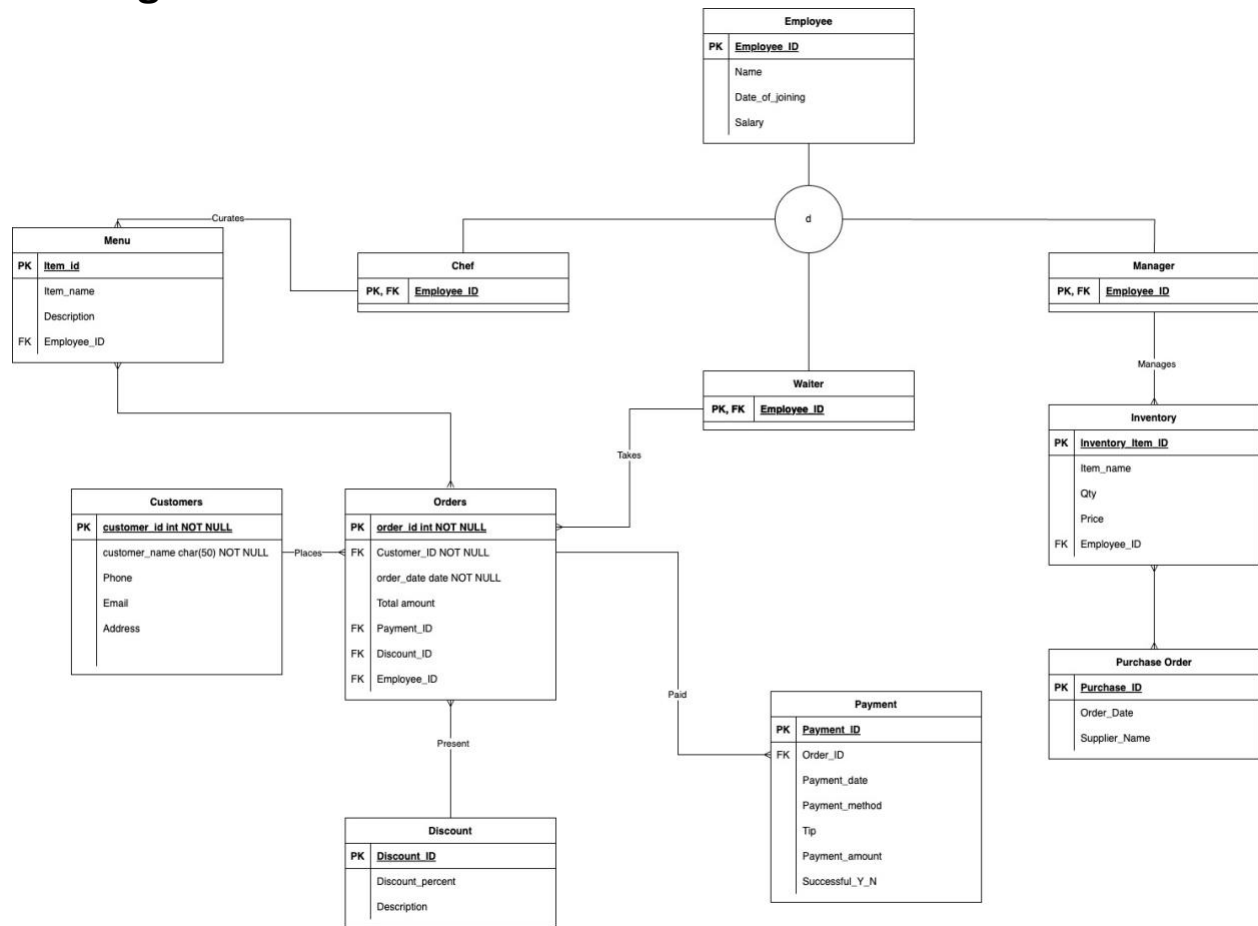
Percentage of Effort Contributed by Giridhar Babu: 50%

Signature of Student 1: Ankur Pandey

Signature of Student 2: Giridhar Babu

Submission Date: 10/04/ 2024

ER Diagram



Entities:

1. **Customer:** Represents customers who place orders.
 - Attributes: customer_id (PK), customer_name, phone, email, address
2. **Order:** Represents food orders made by customers.
 - Attributes: order_id (PK), order_date, total_amount, customer_id (FK), payment_id, employee_id (FK), discount_id (FK)
3. **Payment:** Represents payments made for orders.
 - Attributes: payment_id (PK), order_id (FK), payment_date, payment_type, amount
4. **Discount:** Represents discounts available for orders.
 - Attributes: discount_id (PK), discount_percentage, description
5. **Menu:** Contains all food items available.
 - Attributes: item_id (PK), item_name, price, employee_id (FK)
6. **Employee:** Represents employees, which include Waitrs, Managers, and Chefs.
 - Attributes: employee_id (PK), name, Date_of_joining, salary
7. **Waiter** (Specializes Employee):
 - Attributes: employee_id (PK, FK)
8. **Chef** (Specializes Employee):
 - Attributes: employee_id (PK, FK), specialization, years_of_experience
9. **Manager** (Specializes Employee):
 - Attributes: employee_id (PK, FK)
10. **Inventory:** Represents the food inventory.
 - Attributes: inventory_item_id (PK), item_name, qty, unit_price, Employee_id
11. **Purchase Order:** Represents purchase orders for inventory.
 - Attributes: purchase_id (PK), inventory_id (FK), order_date, quantity_ordered, supplier_name

Relationships:

1. Customer to Order:

- A customer can place multiple orders, but each order is placed by a single customer.
- Relationship: Customer (1) → Order (N)
- Type: One-to-Many (1)
- Foreign Key: customer_id in the Order table refers to customer_id in the Customer table.

2. Order to Payment:

- Each order can have one or more payments, but each payment is for a specific order.
- Relationship: Order (1) → Payment (N)
- Type: One-to-Many (1)
- Foreign Key: order_id in the Payment table refers to order_id in the Order table.

3. Order to Discount:

- An order can apply a discount, but a discount can be used for multiple orders.
- Relationship: Order (N) → Discount (1)
- Type: Many-to-One (N:1)
- Foreign Key: discount_id in the Order table refers to discount_id in the Discount table.

4. Order to Menu (via Junction Table Order_Menu):

- An order can contain multiple menu items, and a menu item can be part of multiple orders (many-to-many relationship).
- Relationship: Order (N) → Order_Menu ← Menu (N)
- Type: Many-to-Many (N)
- Junction Table: Order_Menu with order_id (FK) and menu_id (FK) links Order and Menu.

5. Employee to Order (for Waiters):

- A waiter can serve multiple orders, but an order is served by a single employee (waiter).
- Relationship: Employee (Waiter) (1) → Order (N)
- Type: One-to-Many (1)
- Foreign Key: employee_id in the Order table refers to employee_id in the Employee table for Waiters.

6. Employee to Menu (for Chefs):

- A chef can create multiple menu items, but a menu item is associated with a single chef.
- Relationship: Employee (Chef) (1) → Menu (N)
- Type: One-to-Many (1)
- Foreign Key: employee_id in the Menu table refers to employee_id in the Employee table for Chefs.

7. Employee to Inventory (for Managers):

- A manager can oversee multiple inventory items, but each inventory item is associated with one manager.
- Relationship: Employee (Manager) (1) → Inventory (N)
- Type: One-to-Many (1)
- Foreign Key: employee_id in the Inventory table refers to employee_id in the Employee table for Managers.

8. Purchase Order to Inventory:

- Each purchase order is made for a single inventory item, but each inventory item can have multiple purchase orders over time.
- Relationship: Inventory (1) → Purchase Order (N)
- Type: One-to-Many (1)
- Foreign Key: inventory_item_id in the Purchase Order table refers to inventory_item_id in the Inventory table.

9. Employee Specialization (Inheritance):

- Employee generalizes Waiter, Chef, and Manager. Each employee can be specialized into one of these roles.
- Type: One-to-One (1:1) Inheritance
- Foreign Key: employee_id is shared across Employee, Waiter, Chef, and Manager.

Diagram Structure:

- Customer → Order → Payment, Discount
- Order → Menu (N relationship)
- Menu → Category
- Order → Waitress (Handled by waitress from the Employee table)
- Manager → Employee
- Inventory → Purchase Order

Dimensions:

1. Customer Dimension

- **Attributes:** customer_id, customer_name, phone, email, address
- **Hierarchy:** None

2. Order Dimension

- **Attributes:** order_id, order_date, total_amount, customer_id, payment_id, employee_id, discount_id
- **Hierarchy:** order_date (can be split into year > month > day for time-based analysis)

3. Employee Dimension

- **Attributes:** employee_id, name, date_of_joining, salary
- **Hierarchy:** None

4. Menu Dimension

- **Attributes:** item_id, item_name, price, employee_id
- **Hierarchy:** Menu Type (Veg/Non-Veg) > Item Name

5. Payment Dimension

- **Attributes:** payment_id, order_id, payment_date, payment_type, amount
- **Hierarchy:** payment_date (can be split into year > month > day for time-based analysis)

6. Discount Dimension

- **Attributes:** discount_id, discount_percentage, description
- **Hierarchy:** None

7. Inventory Dimension

- **Attributes:** inventory_item_id, item_name, qty, unit_price, employee_id
- **Hierarchy:** None

8. Purchase Order Dimension

- **Attributes:** purchase_id, inventory_item_id, order_date, quantity_ordered, supplier_name
- **Hierarchy:** order_date (can be split into year > month > day for time-based analysis)

Measures

1. Total Bill Amount

- **Source:** Order.total_amount
- **Type:** Additive
- **Aggregation:** SUM
- **Description:** Represents the total value of orders placed by customers. This can be aggregated across various dimensions such as time, employee (waiter/chef), customer, and menu items.

2. Total Payment Amount

- **Source:** Payment.amount
- **Type:** Additive
- **Aggregation:** SUM
- **Description:** Represents the total payments made by customers. This measure provides insights into revenue collected across dimensions like time, payment type, and order.

3. Discount Amount

- **Source:** Discount.discount_percentage * Order.total_amount
- **Type:** Calculated
- **Aggregation:** SUM
- **Description:** Represents the total discount applied to orders. This helps analyze the impact of discounts on revenue.

4. Total Inventory Cost

- **Source:** Inventory.unit_price * Inventory.qty
- **Type:** Additive
- **Aggregation:** SUM
- **Description:** Represents the total cost of inventory items. It can be aggregated by item or supplier to understand inventory costs.

5. Total Items Ordered

- **Source:** Count of Order_items (assuming a table links each order to items)
- **Type:** Additive
- **Aggregation:** COUNT
- **Description:** This measure represents the total number of items ordered. It can be analyzed across time, customers, and menu items to understand purchasing trends.

6. Number of Orders

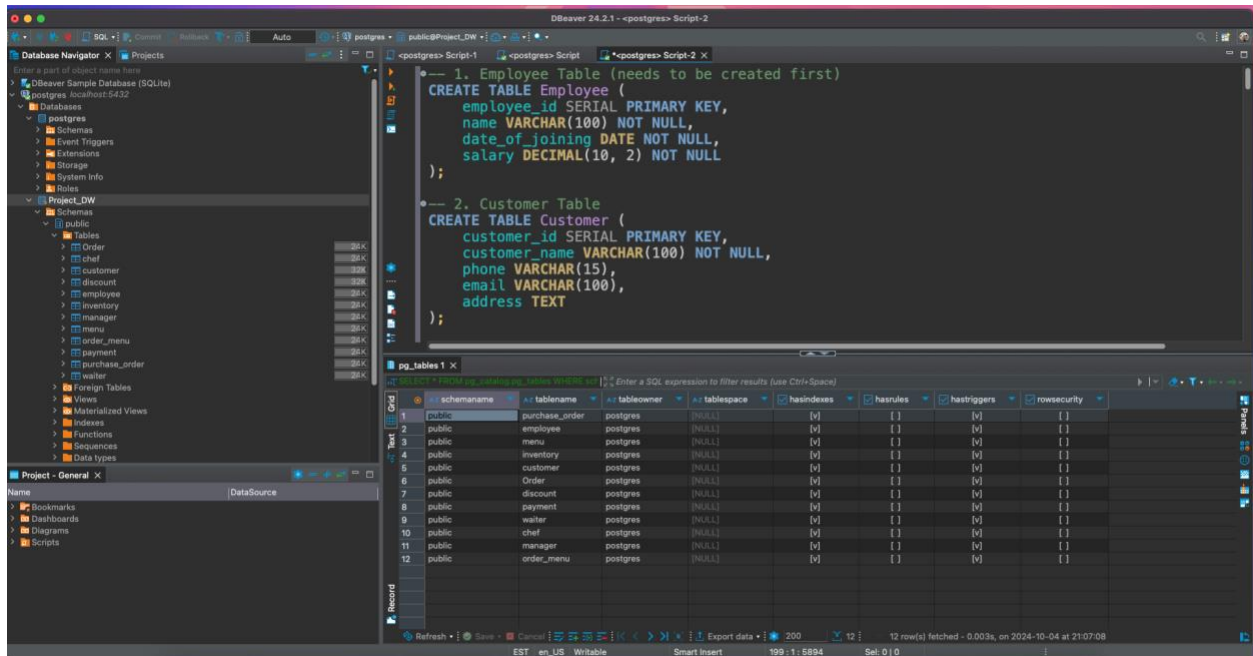
- **Source:** Order.order_id
- **Type:** Count
- **Aggregation:** COUNT
- **Description:** Represents the total number of orders placed. It can be broken down by time, customer, or employee (waiter/manager).

SQL Query Execution Order Summary:

1. **Employee**
2. **Customer**
3. **Discount**
4. **Menu** (references **Employee**)
5. **Inventory** (references **Employee**)
6. **Purchase_Order** (references **Inventory**)
7. **Order** (references **Customer**, **Employee**, and **Discount**)
8. **Payment** (references **Order**)
9. **Add payment_id foreign key constraint to Order**
10. **Waiter** (specializes **Employee**)
11. **Chef** (specializes **Employee**)
12. **Manager** (specializes **Employee**)
13. **Order_Menu** (junction table between **Order** and **Menu**)

Implementation in SQL:

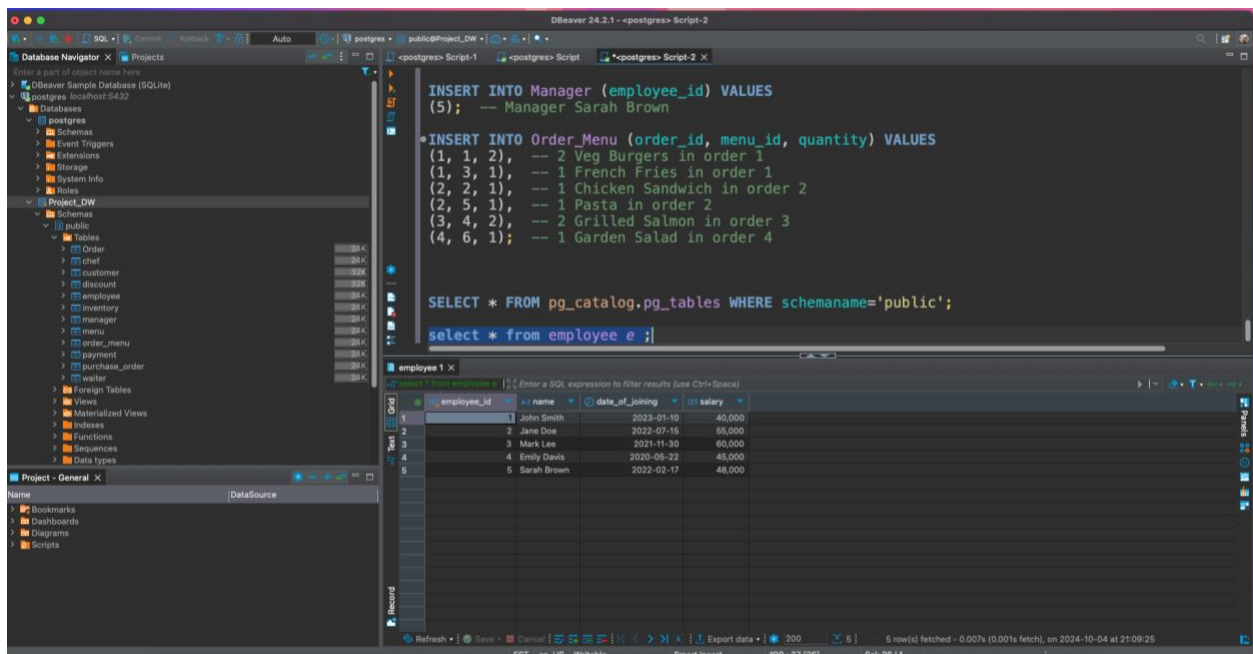
1. List of all the tables being created



The screenshot shows the DBeaver SQL editor with two SQL scripts. The first script creates the 'Employee' table with columns: employee_id (SERIAL PRIMARY KEY), name (VARCHAR(100) NOT NULL), date_of_joining (DATE NOT NULL), and salary (DECIMAL(10, 2) NOT NULL). The second script creates the 'Customer' table with columns: customer_id (SERIAL PRIMARY KEY), customer_name (VARCHAR(100) NOT NULL), phone (VARCHAR(15)), email (VARCHAR(100)), and address (TEXT). Below the scripts, a table named 'pg_tables' is displayed, listing all tables in the 'public' schema.

tableid	schemaname	tablename	tableowner	tablespace	hasindexes	hasrules	hastiggers	rowsecurity
1	public	purchase_order	postgres	[NULL]	[v]	[f]	[v]	[f]
2	public	employee	postgres	[NULL]	[v]	[f]	[v]	[f]
3	public	menu	postgres	[NULL]	[v]	[f]	[v]	[f]
4	public	inventory	postgres	[NULL]	[v]	[f]	[v]	[f]
5	public	customer	postgres	[NULL]	[v]	[f]	[v]	[f]
6	public	Order	postgres	[NULL]	[v]	[f]	[v]	[f]
7	public	discount	postgres	[NULL]	[v]	[f]	[v]	[f]
8	public	payment	postgres	[NULL]	[v]	[f]	[v]	[f]
9	public	walter	postgres	[NULL]	[v]	[f]	[v]	[f]
10	public	chef	postgres	[NULL]	[v]	[f]	[v]	[f]
11	public	manager	postgres	[NULL]	[v]	[f]	[v]	[f]
12	public	order_menu	postgres	[NULL]	[v]	[f]	[v]	[f]

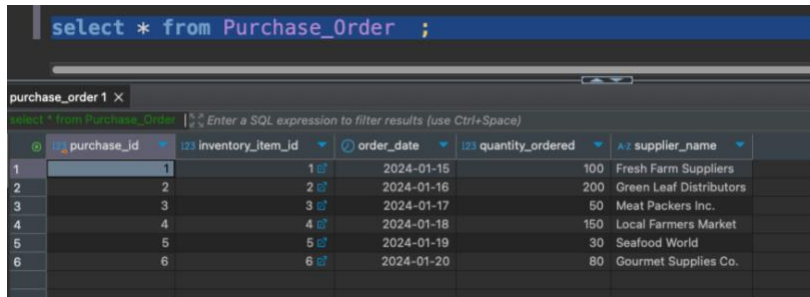
2. List of values being populated in the Employee table



The screenshot shows the DBeaver SQL editor with two SQL scripts. The first script inserts data into the 'Manager' table (employee_id 5) and the 'Order_Menu' table. The second script selects all data from the 'employee' table. Below the scripts, a table named 'employee' is displayed, showing the inserted data.

employee_id	name	date_of_joining	salary
1	John Smith	2023-01-10	40,000
2	Jane Doe	2022-07-15	55,000
3	Mark Lee	2021-11-30	60,000
4	Emily Davis	2020-05-22	45,000
5	Sarah Brown	2022-02-17	48,000

3. List of values being populated in the Purchase_order table



The screenshot shows a database query interface. At the top, a SQL query is entered: `select * from Purchase_Order ;`. Below the query, the results are displayed in a table. The table has six columns: `purchase_id`, `inventory_item_id`, `order_date`, `quantity_ordered`, and `supplier_name`. The results show six rows of data, each representing a purchase order.

	purchase_id	inventory_item_id	order_date	quantity_ordered	supplier_name
1	1	1	2024-01-15	100	Fresh Farm Suppliers
2	2	2	2024-01-16	200	Green Leaf Distributors
3	3	3	2024-01-17	50	Meat Packers Inc.
4	4	4	2024-01-18	150	Local Farmers Market
5	5	5	2024-01-19	30	Seafood World
6	6	6	2024-01-20	80	Gourmet Supplies Co.

Similarly, we have populated all our tables. For now, we have inserted only 5 records in each table, but we aim to add up to 1,000 records by the next milestone. I'm attaching the SQL script in a .txt file format below.

https://drive.google.com/file/d/1GDqnbHKxYPCRfLZLQZl4awitcco_hprH/view?usp=sharing

Data Population Methodology:

We generated our data in batches using various LLMs and first populated the static data, such as Employee, Customer, and Items. Using this static data, we then generated the transactional data, such as Amount paid, etc.