

2023

SQL Database Design and Implementation

DEEPANKAR RIJAL

Executive Summary

A customer uses the Manoosh Pizzeria's online ordering system to customise their pizza order, choose delivery, and pay for it in order to get ready for a gathering. The pizza ordering system is represented by 13 entities in the ERD, which are linked together for data management by relationships and attributes. Primary and foreign keys are used in the database design to preserve data consistency and avoid duplication, which makes it easier to track orders, customise them, offer promotions, and handle deliveries effectively.

Here are the key findings of the report:

- Data Consistency: Primary and foreign keys help to maintain data consistency.
- Order Tracking: Order tracking is enabled, and it is thorough.
- Customization: Customers have the option to customise orders.
- Promotions: Specific orders may be linked to promotional deals.
- Delivery Control: Effective delivery control is encouraged.
- Employee structure: Manager and employee entities specify the organisational structure.

Table of Contents

<i>Executive Summary</i>	1
1. ER Model	5
1.1 Scenario: An online pizza ordering system	5
1.2 Initial Entities	5
1.3 Normalization	6
1.4 Final Entities.....	7
1.5 ER Diagram.....	9
2. pgAdmin Screenshot	9
3. DDL Statements	10
3.1 customer.....	10
3.2 promotion.....	11
3.3 order.....	12
3.4 payment.....	13
3.5 ingredient	14
3.6 topping	15
3.7 pizza	16
3.8 pizzatopping.....	17
3.9 pizzaingredient.....	18
3.10 branch.....	19
3.11 employee	20
3.12 delivery_driver	21
3.13 manager.....	22
4. DML Statements	23
customer	23
promotion	24
order	25
payment.....	26
ingredient.....	27
topping.....	28
pizza	29
pizzatopping	30
pizzaingredient	31
branch	32
employee.....	33

delivery_driver	34
manager	35
5. Queries and Results.....	36
Query 1: Using INNER JOIN.....	36
Query 2: Using OUTER JOIN.....	37
Query 3: Using Specialization Hierarchy	38
Query 4: Using GROUP BY and HAVING	39
Query 5: Using INNER SQL Query.....	40
Query 6: Using UPDATE	41
References.....	42

Figure 1 3NF Normalization	7
Figure 2 ERD Model.....	9
Figure 3 Database Creation Image 1	9
Figure 4 Database Creation Image 2	10
Figure 5 DDL for customer Table.....	11
Figure 6 DDL for promotion Table	12
Figure 7 DDL for order Table	13
Figure 8 DDL for payment Table.....	14
Figure 9 DDL for ingredient Table	15
Figure 10 DDL for topping Table.....	16
Figure 11 DDL for pizza Table	17
Figure 12 DDL for pizzatopping Table.....	18
Figure 13 DDL for pizzaingredient Table	19
Figure 14 DDL for branch Table.....	20
Figure 15 DDL for employee Table	21
Figure 16 DDL for delivery_driver Table.....	22
Figure 17 DDL for manager Table.....	23
Figure 18 DML for customer Table.....	24
Figure 19 DML for promotion Table.....	25
Figure 20 DML for order Table	26
Figure 21 DML for payment Table.....	27
Figure 22 DML for ingredient Table	28
Figure 23 DML for topping Table.....	29
Figure 24 DML for pizza Table	30
Figure 25 DML for pizzatopping Table.....	31
Figure 26 DML for pizzaingredient Table	32
Figure 27 DML for branch Table	33
Figure 28 DML for employee Table	34
Figure 29 DML for delivery_driver Table.....	35
Figure 30 DML for manager Table.....	36
Figure 31 Query 1.....	37
Figure 32 Query 2.....	38

Figure 33 Query 3.....	39
Figure 34 Query 4.....	40
Figure 35 Query 5.....	41
Figure 36 Query 6.....	42

1. ER Model

1.1 Scenario: An online pizza ordering system

In order to prepare for a gathering, a customer chooses to place an online order for a mouth-watering spread from Manoosh Pizzeria. They begin their culinary adventure by creating an account as a devoted customer on the Manoosh Pizzeria website. They select the substantial "huge" size to ensure there is plenty for everyone, choosing to savour the flavours of both the legendary "Super Supreme" and the classic "Margherita" pizzas.

Our customer creates their own unique pizzas by adding a variety of delectable toppings and well-chosen ingredients to the culinary canvas, creating a unique pizza experience. They choose the delivery option and carefully enter their address, leaving no possibility for delivery confusion, in order to make the evening hassle-free.

They head to the order summary page with their flawless pizzas in the virtual cart. Here, they determine the whole price by taking into account the base prices, alluring customizations, and small delivery costs. They confidently make their way to the cashier after being satisfied with their choices and the price.

They decide to use a credit card payment option because it is convenient at this key stage. They pay the invoice after safely entering their credit card information, completing the order process. Once the money has been accepted, an order is generated with a special order ID.

The order is quickly forwarded to the closest Manoosh Pizzeria location, where the capable branch manager receives it. The branch manager quickly assigns the duty to a devoted delivery driver who is in charge of making sure that this culinary treasure arrives on time.

The delivery driver gets ready for the trip as the pizzas are painstakingly made in the restaurant's kitchen. Armed with the customer's address and the order information, they make their way through streets and traffic determined to deliver the pizzas on schedule and in perfect shape.

As the delivery driver approaches the customer's doorstep, the decisive moment occurs. They hand the highly anticipated consumer a hot pizza after double-checking the delivery address. The consumer takes the purchase with a delighted smile, signalling the successful conclusion of an easy online ordering process.

1.2 Initial Entities

1. Customer:

- Attributes:
cust_id (Primary Key), **cust_fname**, **cust_lname**, **cust_address**, **cust_phone**, **cust_email**

2. Order:

- Attributes:

order_id (Primary Key), order_date, total_amount, *cust_id* (Foreign Key),
promotion_id (Foreign Key)

3. **Pizza:**

- Attributes:
pizza_id (Primary Key), pizza_name, description, price, topping_id, top_name,
top_price, ingredient_id, ing_name, ing_price, *order_id* (Foreign Key)

4. **Payment:**

- Attributes:
payment_id (Primary Key), pay_date, amount, pay_method, *order_id*
(Foreign Key)

5. **delivery_driver:**

- Attributes:
emp_id (Primary Key), license_num, *order* (Foreign Key)

6. **branch:**

- Attributes:
branch_id (Primary Key), branch_name, location

7. **employee:**

- Attributes:
emp_id (Primary Key), emp_fname, emp_lname, position, salary,
emp_phone, *branch_id* (Foreign Key)

8. **manager:**

- Attributes:
emp_id (Primary Key), emp_fname, emp_lname, join_date

9. **promotion:**

- Attributes:
promotion_id (Primary Key), promo_name, description, discount_percent,
start_date, end_date

1.3 Normalization

1NF

All the initial entities are in 1NF as:

- There exists primary key in every entities.
- There is no multivalued attributes

2NF

All the initial entities are in 2NF as there is no existence of partial dependency in any table.

3NF

All the initial entities are in 3NF except pizza as there is no existence of transitive dependency.

For pizza entity, we normalise the table into 3NF as shown in the picture:

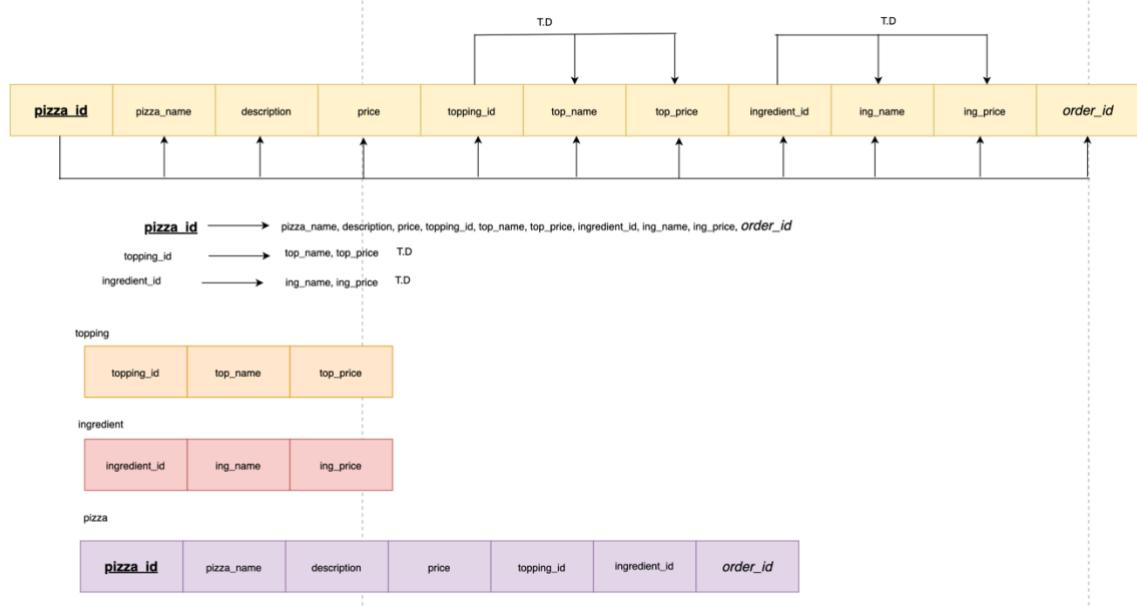


Figure 1 3NF Normalization

Here, 3 tables are created out of 1 pizza table after normalization into 3NF.

1.4 Final Entities

In the final entities, I have categorized branch as supertype and delivery_driver and manager are subtypes of branch entity. Also, delivery_driver and manager are overlapping subsets (Zainurrohman, 2021).

1. customer:

- Attributes:
cust_id (Primary Key), cust_fname, cust_lname, cust_address, cust_phone, cust_email

2. promotion:

- Attributes:
promotion_id (Primary Key), promo_name, description, discount_percent, start_date, end_date

3. order:

- Attributes:
order_id (Primary Key), order_date, total_amount, *cust_id* (Foreign Key), *promotion_id* (Foreign Key)

4. payment:

- Attributes:
payment_id (Primary Key), pay_date, amount, pay_method, *order_id* (Foreign Key)

5. **topping:**
 - Attributes:
topping_id (Primary Key), top_name, top_price

6. **ingredient:**
 - Attributes:
ingredient_id (Primary Key), ing_name, ing_price

7. **pizza:**
 - Attributes:
pizza_id (Primary Key), pizza_name, description, price, *topping_id* (Foreign Key), *ingredient_id* (Foreign Key), *order_id* (Foreign Key)

8. **pizzatopping:**
 - Attributes:
Pizzatopping_id (Surrogate Primary Key), *topping_id* (Foreign Key), *pizza_id* (Foreign Key)

9. **pizzaingredient:**
 - Attributes:
Pizzaingredient_id (Surrogate Primary Key), *ingredient_id* (Foreign Key), *pizza_id* (Foreign Key)

10. **delivery_driver:**
 - Attributes:
emp_id (Primary Key), license_num, *order* (Foreign Key)

11. **branch:**
 - Attributes:
branch_id (Primary Key), branch_name, location

12. **employee:**
 - Attributes:
emp_id (Primary Key), emp_fname, emp_lname, position, salary, emp_phone, *branch_id* (Foreign Key)

13. **manager:**
 - Attributes:
emp_id (Primary Key), emp_fname, emp_lname, join_date

Composite key:

Pizza_id and topping_id (Composite key) as two foreign keys together form composite key
 Pizza_id and ingredient_id (Composite key) as two foreign keys together form composite key

Surrogate key: pizzatopping_id, pizzaingredient_id

1.5 ER Diagram

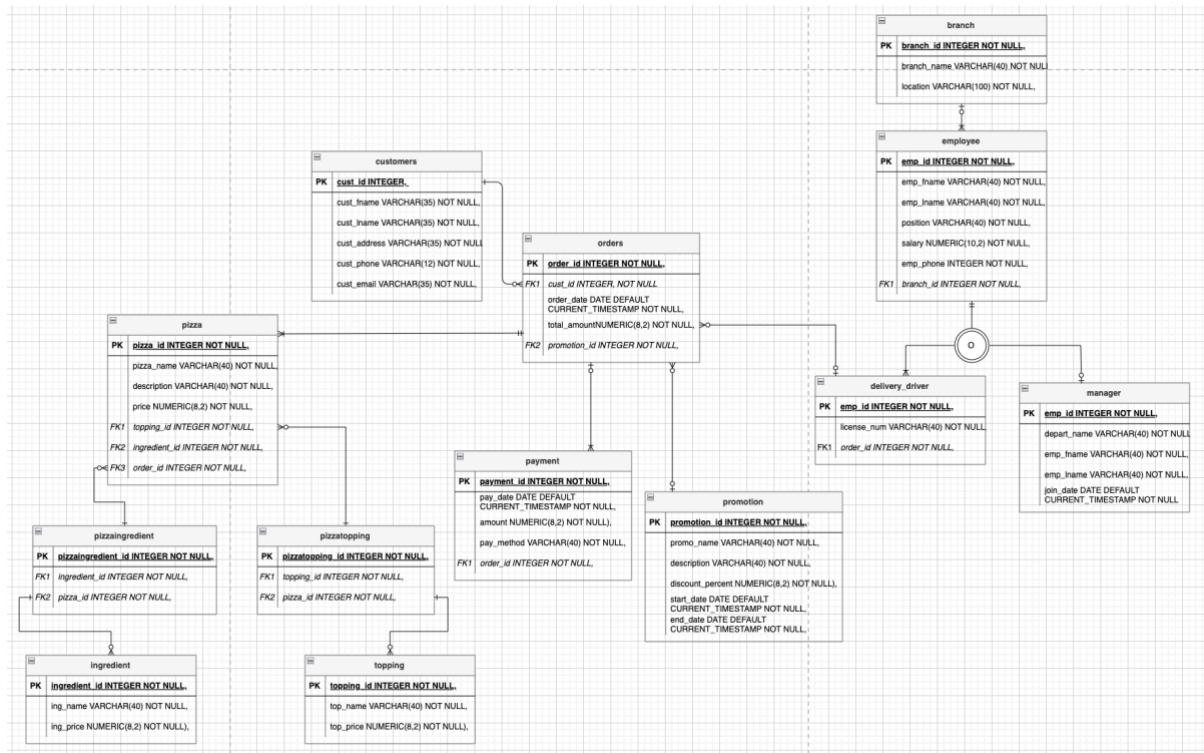


Figure 2 ERD Model

2. pgAdmin Screenshot

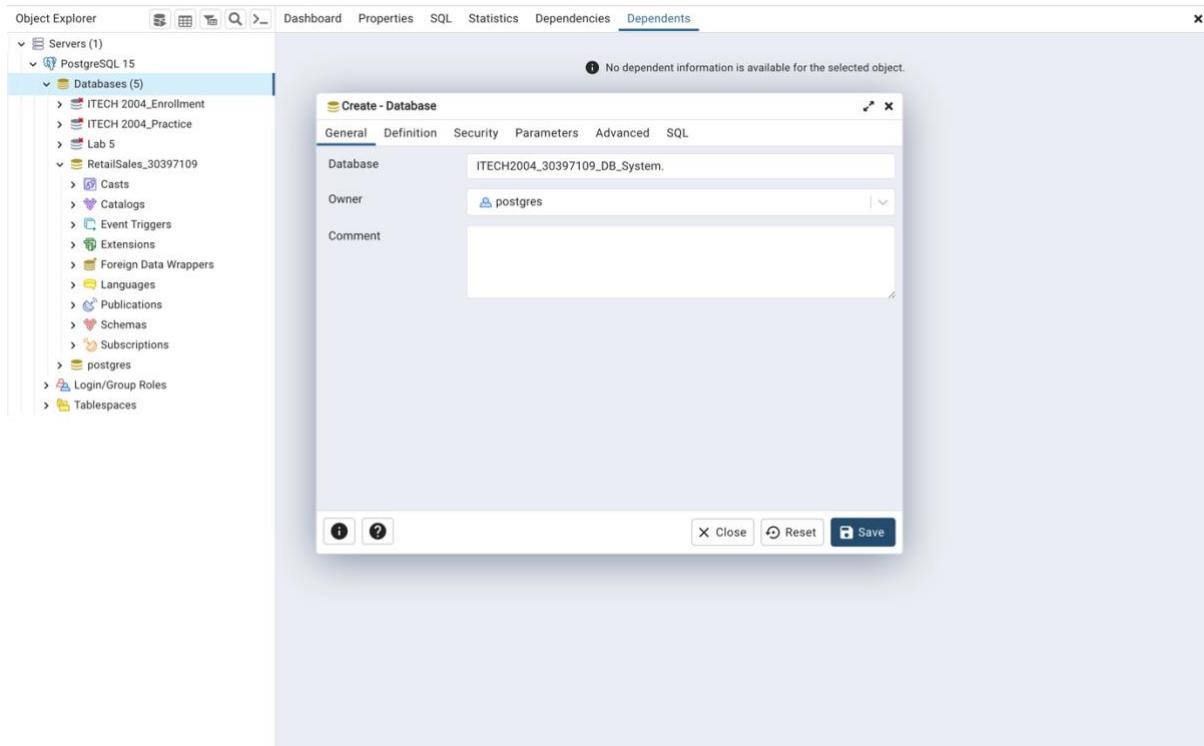


Figure 3 Database Creation Image 1

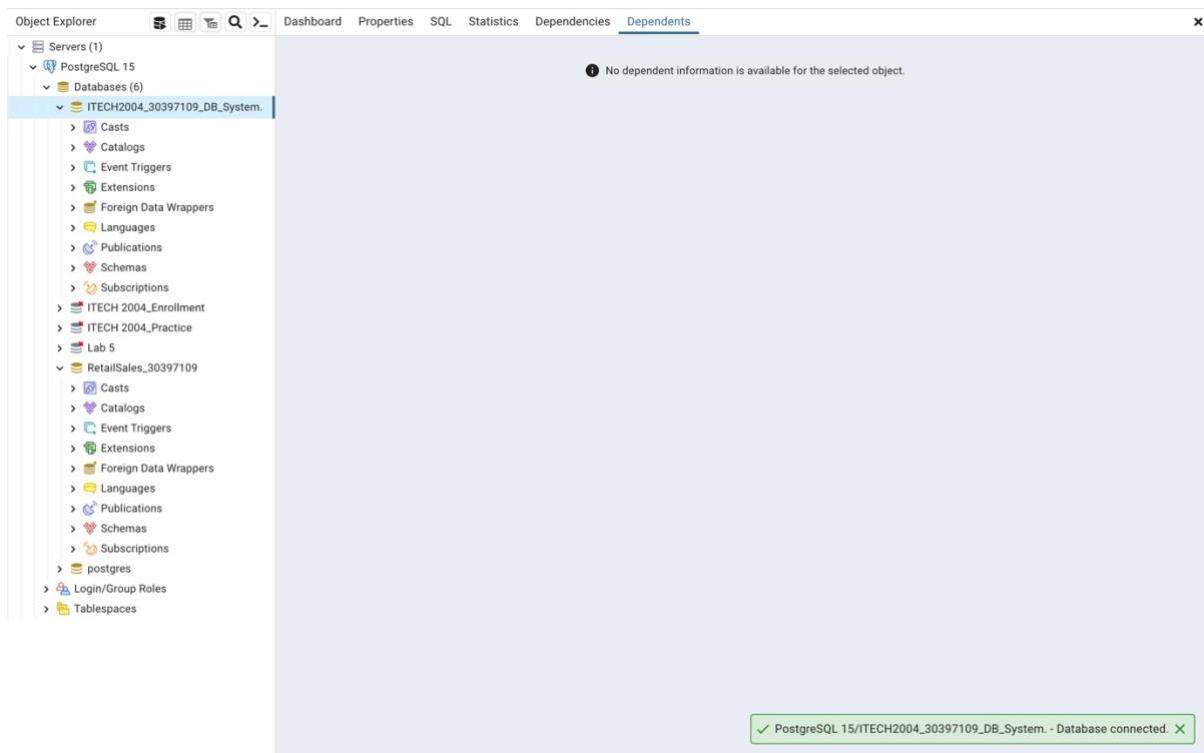


Figure 4 Database Creation Image 2

3. DDL Statements

3.1 customer

START TRANSACTION;

```
CREATE TABLE customer (
    cust_id          INTEGER,
    cust_fname       VARCHAR(35) NOT NULL,
    cust_lname       VARCHAR(35) NOT NULL,
    cust_address     VARCHAR(35) NOT NULL,
    cust_phone       VARCHAR(12) NOT NULL,
    cust_email       VARCHAR(35) NOT NULL,
    PRIMARY KEY (cust_id),
    CONSTRAINT cus_ui1 UNIQUE(cust_fname, cust_lname, cust_phone));
```

COMMIT;

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying a tree view of databases, schemas, and objects. The right pane is the Query Editor, showing a SQL script for creating a 'customer' table. The script includes a transaction block, column definitions, and a unique constraint. The status bar at the bottom indicates the query was completed successfully in 43 msec.

```

1 START TRANSACTION;
2
3 CREATE TABLE customer (
4     cust_id      INTEGER,
5     cust_fname   VARCHAR(35) NOT NULL,
6     cust_lname   VARCHAR(35) NOT NULL,
7     cust_address VARCHAR(35) NOT NULL,
8     cust_phone   VARCHAR(12) NOT NULL,
9     cust_email   VARCHAR(35) NOT NULL,
10    PRIMARY KEY (cust_id),
11    CONSTRAINT cus_u1 UNIQUE(cust_fname, cust_lname, cust_phone));
12
13 COMMIT;

```

Total rows: 0 of 0 Query complete 00:00:00.043 ✓ Query returned successfully in 43 msec. X
Ln 13, Col 1

Figure 5 DDL for customer Table

3.2 promotion

START TRANSACTION;

```

CREATE TABLE promotion (
promotion_id      INTEGER NOT NULL,
promo_name       VARCHAR(40) NOT NULL,
description      VARCHAR(40) NOT NULL,
discount_percent NUMERIC(8,2) NOT NULL,
start_date       DATE DEFAULT CURRENT_TIMESTAMP NOT NULL,
end_date         DATE DEFAULT CURRENT_TIMESTAMP NOT NULL,
PRIMARY KEY (promotion_id));

```

COMMIT;

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying the database structure. The right pane is the Query Editor, showing the SQL code for creating the promotion table.

```

1 START TRANSACTION;
2
3 CREATE TABLE promotion (
4     promotion_id      INTEGER NOT NULL,
5     promo_name        VARCHAR(49) NOT NULL,
6     description        VARCHAR(49) NOT NULL,
7     discount_percent   NUMERIC(8,2) NOT NULL,
8     start_date        DATE DEFAULT CURRENT_TIMESTAMP NOT NULL,
9     end_date          DATE DEFAULT CURRENT_TIMESTAMP NOT NULL,
10    PRIMARY KEY (promotion_id));
11
12 COMMIT;

```

The Query Editor also displays the message "Query returned successfully in 43 msec." and "Ln 12, Col 8".

Figure 6 DDL for promotion Table

3.3 order

START TRANSACTION;

```

CREATE TABLE "order" (
    order_id          INTEGER NOT NULL,
    order_date        DATE DEFAULT CURRENT_TIMESTAMP NOT NULL,
    total_amount      NUMERIC(8,2) NOT NULL,
    cust_id           INTEGER,
    promotion_id      INTEGER NOT NULL,
    PRIMARY KEY (order_id),
    FOREIGN KEY (cust_id) REFERENCES customer ON DELETE CASCADE,
    FOREIGN KEY (promotion_id) REFERENCES promotion(promotion_id));

```

COMMIT;

```

1 START TRANSACTION;
2
3 CREATE TABLE "order" (
4     order_id      INTEGER NOT NULL,
5     order_date    DATE DEFAULT CURRENT_TIMESTAMP NOT NULL,
6     total_amount  NUMERIC(8,2) NOT NULL,
7     cust_id       INTEGER,
8     promotion_id INTEGER NOT NULL,
9     PRIMARY KEY (order_id),
10    FOREIGN KEY (cust_id) REFERENCES customer ON DELETE CASCADE,
11    FOREIGN KEY (promotion_id) REFERENCES promotion(promotion_id));
12
13 COMMIT;
14

```

Query returned successfully in 123 msec.

Figure 7 DDL for order Table

3.4 payment

START TRANSACTION;

```

CREATE TABLE payment (
payment_id      INTEGER NOT NULL,
pay_date        DATE DEFAULT CURRENT_TIMESTAMP NOT NULL,
amount          NUMERIC(8,2) NOT NULL,
pay_method      VARCHAR(40) NOT NULL,
order_id        INTEGER NOT NULL,
PRIMARY KEY (payment_id),
FOREIGN KEY (order_id) REFERENCES "order"(order_id));

```

COMMIT;

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer tree, which includes Schemas (1), public, Tables (3), customer, order, promotion, and several system schemas. The Tables node is expanded. In the center is the Query Editor window with the following SQL code:

```

1 START TRANSACTION;
2
3 CREATE TABLE payment (
4     payment_id      INTEGER NOT NULL,
5     pay_date        DATE DEFAULT CURRENT_TIMESTAMP NOT NULL,
6     amount          NUMERIC(8,2) NOT NULL,
7     pay_method      VARCHAR(40) NOT NULL,
8     order_id        INTEGER NOT NULL,
9     PRIMARY KEY (payment_id),
10    FOREIGN KEY (order_id) REFERENCES "order"(order_id);
11
12 COMMIT;
13

```

Below the query editor is the Data Output tab, which displays the message: "Query returned successfully in 45 msec." At the bottom of the screen, a status bar shows "Total rows: 0 of 0" and "Query complete 00:00:00.045". A green success message in the status bar says "Query returned successfully in 45 msec. [X] Ln 7, Col 34".

Figure 8 DDL for payment Table

3.5 ingredient

START TRANSACTION;

```

CREATE TABLE ingredient (
ingredient_id      INTEGER NOT NULL,
ing_name          VARCHAR(40) NOT NULL,
ing_price         NUMERIC(8,2) NOT NULL,
PRIMARY KEY (ingredient_id));

```

COMMENT;

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer pane, which lists various database objects like Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas, and Tables. The Tables section is expanded, showing customer, order, payment, promotion, Trigger Functions, Types, Views, Subscriptions, and several system tables (ITECH 2004_Enrollment, ITECH 2004_Practice, Lab 5, lab 7, postgres). The main pane contains a SQL query editor with the following DDL code:

```

1 START TRANSACTION;
2
3 CREATE TABLE ingredient (
4     ingredient_id      INTEGER NOT NULL,
5     ing_name           VARCHAR(40) NOT NULL,
6     ing_price          NUMERIC(8,2) NOT NULL,
7     PRIMARY KEY (ingredient_id);
8
9 COMMIT;
10

```

Below the query editor, the Data Output tab is selected, showing the message "Query returned successfully in 41 msec." and the status bar indicating "Total rows: 0 of 0" and "Query complete 00:00:00.041". A green success message at the bottom right says "Query returned successfully in 41 msec. Ln 6, Col 33".

Figure 9 DDL for ingredient Table

3.6 topping

START TRANSACTION;

```

CREATE TABLE topping (
    topping_id      INTEGER NOT NULL,
    top_name        VARCHAR(40) NOT NULL,
    top_price       NUMERIC(8,2) NOT NULL,
    PRIMARY KEY (topping_id));

```

COMMENT;

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Object Explorer' and lists various database objects like Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas, and Tables. The 'Tables' section is expanded, showing customer, order, payment, promotion, Trigger Functions, Types, Views, Subscriptions, and several schema entries (ITECH 2004_Enrollment, ITECH 2004_Practice, Lab 5, lab 7, postgres). The main pane is titled 'Query' and contains the following SQL code:

```

1 START TRANSACTION;
2
3 CREATE TABLE topping (
4     topping_id      INTEGER NOT NULL,
5     top_name        VARCHAR(40) NOT NULL,
6     top_price       NUMERIC(8,2) NOT NULL,
7     PRIMARY KEY (topping_id);
8
9 COMMIT;
10

```

Below the code, the 'Data Output' tab is selected, showing the message 'Query returned successfully in 90 msec.' and the status bar indicating 'Total rows: 0 of 0 Query complete 00:00:00.090'. A green success message at the bottom right says 'Query returned successfully in 90 msec. Ln 6, Col 33'.

Figure 10 DDL for topping Table

3.7 pizza

START TRANSACTION;

```

CREATE TABLE pizza (
    pizza_id      INTEGER NOT NULL,
    pizza_name    VARCHAR(40) NOT NULL,
    description   VARCHAR(40) NOT NULL,
    price         NUMERIC(8,2) NOT NULL,
    topping_id    INTEGER NOT NULL,
    ingredient_id INTEGER NOT NULL,
    order_id      INTEGER NOT NULL,
    PRIMARY KEY (pizza_id),
    FOREIGN KEY (order_id) REFERENCES "order" ON DELETE CASCADE,
    FOREIGN KEY (ingredient_id) REFERENCES ingredient(ingredient_id),
    FOREIGN KEY (topping_id) REFERENCES topping(topping_id));

```

COMMIT;

```

1 START TRANSACTION;
2
3 CREATE TABLE pizza (
4     pizza_id      INTEGER NOT NULL,
5     pizza_name    VARCHAR(48) NOT NULL,
6     description   VARCHAR(48) NOT NULL,
7     price         NUMERIC(8,2) NOT NULL,
8     topping_id    INTEGER NOT NULL,
9     ingredient_id INTEGER NOT NULL,
10    order_id      INTEGER NOT NULL,
11    PRIMARY KEY (pizza_id),
12    FOREIGN KEY (order_id) REFERENCES "order" ON DELETE CASCADE,
13    FOREIGN KEY (ingredient_id) REFERENCES ingredient(ingredient_id),
14    FOREIGN KEY (topping_id) REFERENCES topping(topping_id);
15
16 COMMIT;
17

```

Object Explorer Statistics Dependencies Dependents lab 7/postgres... ITECH2004_30397109_DB_System./postgres@PostgreSQL 15*

Query Query History Scratch Pad

Data Output Messages Notifications

Commit
Query returned successfully in 37 msec.

Total rows: 0 of 0 Query complete 00:00:00.037 ✓ Query returned successfully in 37 msec. X Ln 12, Col 36

Figure 11 DDL for pizza Table

3.8 pizzatopping

START TRANSACTION;

```

CREATE TABLE pizzatopping (
pizzatopping_id          INTEGER NOT NULL,
topping_id                 INTEGER NOT NULL,
pizza_id                  INTEGER NOT NULL,
PRIMARY KEY (pizzatopping_id),
FOREIGN KEY (topping_id) REFERENCES topping(topping_id) ON DELETE CASCADE,
FOREIGN KEY (pizza_id) REFERENCES pizza(pizza_id) ON DELETE CASCADE);

```

COMMIT;

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying various database objects like Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas, and Tables. The right pane is the Query Editor, which contains the following SQL code:

```

1 START TRANSACTION;
2
3 CREATE TABLE pizzatopping (
4     pizzatopping_id      INTEGER NOT NULL,
5     topping_id           INTEGER NOT NULL,
6     pizza_id              INTEGER NOT NULL,
7     PRIMARY KEY (pizzatopping_id),
8     FOREIGN KEY (topping_id) REFERENCES topping(topping_id) ON DELETE CASCADE,
9     FOREIGN KEY (pizza_id) REFERENCES pizza(pizza_id) ON DELETE CASCADE);
10
11 COMMIT;

```

Below the Query Editor, the Data Output tab shows the message "Query returned successfully in 51 msec." and the status bar indicates "Total rows: 0 of 0" and "Query complete 00:00:00.051".

Figure 12 DDL for pizzatopping Table

3.9 pizzaingredient

START TRANSACTION;

```

CREATE TABLE pizzaingredient (
    pizzaingredient_id      INTEGER NOT NULL,
    ingredient_id           INTEGER NOT NULL,
    pizza_id                 INTEGER NOT NULL,
    PRIMARY KEY (pizzaingredient_id),
    FOREIGN KEY (pizza_id) REFERENCES pizza(pizza_id) ON DELETE CASCADE,
    FOREIGN KEY (ingredient_id) REFERENCES ingredient(ingredient_id) ON DELETE CASCADE);

COMMIT;

```

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** On the left, it shows various database objects like Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (1), public (Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables (4)), customer, order, payment, promotion, Trigger Functions, Types, Views, Subscriptions, ITECH 2004_Enrollment, ITECH 2004_Practice, Lab 5, lab 7, postgres, Login/Group Roles, and Tablespaces.
- Query Editor:** The main area contains the following SQL code:


```

1 START TRANSACTION;
2
3 CREATE TABLE pizzaingredient (
4     pizzaingredient_id INTEGER NOT NULL,
5     ingredient_id      INTEGER NOT NULL,
6     pizza_id           INTEGER NOT NULL,
7     PRIMARY KEY (pizzaingredient_id),
8     FOREIGN KEY (pizza_id) REFERENCES pizza(pizza_id) ON DELETE CASCADE,
9     FOREIGN KEY (ingredient_id) REFERENCES ingredient(ingredient_id) ON DELETE CASCADE);
10
11 COMMIT;
      
```
- Data Output:** Below the query editor, it shows the output of the COMMIT command: "Query returned successfully in 40 msec."
- Status Bar:** At the bottom right, it says "Query returned successfully in 40 msec. X Ln 11, Col 8".

Figure 13 DDL for pizzaingredient Table

3.10 branch

START TRANSACTION;

```

CREATE TABLE branch (
branch_id          INTEGER NOT NULL,
branch_name        VARCHAR(40) NOT NULL,
location           VARCHAR(100) NOT NULL,
PRIMARY KEY (branch_id));
      
```

COMMIT;

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer tree, which includes Schemas (1), public (with Tables (4) expanded to show customer, order, payment, promotion, Trigger Functions, Types, Views, Subscriptions, and several system tables like Lab 5, Lab 7, and postgres). The main pane contains a SQL query window with the following DDL code:

```

1 START TRANSACTION;
2
3 CREATE TABLE branch (
4     branch_id      INTEGER NOT NULL,
5     branch_name    VARCHAR(40) NOT NULL,
6     location       VARCHAR(100) NOT NULL,
7     PRIMARY KEY (branch_id));
8
9 COMMIT;

```

Below the query window, the Data Output tab shows the message "Query returned successfully in 45 msec." and the status bar indicates "Total rows: 0 of 0 Query complete 00:00:00.045". A green status bar at the bottom right says "Query returned successfully in 45 msec. Ln 6, Col 9".

Figure 14 DDL for branch Table

3.11 employee

START TRANSACTION;

```

CREATE TABLE employee (
emp_id          INTEGER NOT NULL,
emp_fname        VARCHAR(40) NOT NULL,
emp_lname        VARCHAR(40) NOT NULL,
position         VARCHAR(40) NOT NULL,
salary           NUMERIC(10,2) NOT NULL,
emp_phone        INTEGER NOT NULL,
branch_id        INTEGER NOT NULL,
PRIMARY KEY (emp_id),
FOREIGN KEY (branch_id) REFERENCES branch ON DELETE CASCADE);

```

COMMIT;

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** On the left, it shows the database structure with various objects like Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (1), public (Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables (4)), customer, order, payment, promotion, Trigger Functions, Types, Views, Subscriptions, ITECH 2004_Enrollment, ITECH 2004_Practice, Lab 5, lab 7, postgres, Login/Group Roles, and Tablespaces.
- Query Editor:** The main window contains the following SQL code:


```

1 START TRANSACTION;
2
3 CREATE TABLE employee (
4     emp_id          INTEGER NOT NULL,
5     emp_fname        VARCHAR(40) NOT NULL,
6     emp_lname        VARCHAR(40) NOT NULL,
7     position         VARCHAR(40) NOT NULL,
8     salary           NUMERIC(10,2) NOT NULL,
9     emp_phone        INTEGER NOT NULL,
10    branch_id        INTEGER NOT NULL,
11    PRIMARY KEY (emp_id),
12    FOREIGN KEY (branch_id) REFERENCES branch ON DELETE CASCADE);
13
14 COMMIT;
```
- Data Output:** Below the query editor, it shows the output of the query:

Query returned successfully in 34 msec.
- Messages:** A message bar at the bottom right indicates:

✓ Query returned successfully in 34 msec. X
Ln 12, Col 62

Figure 15 DDL for employee Table

3.12 delivery_driver

START TRANSACTION;

```

CREATE TABLE delivery_driver (
emp_id          INTEGER NOT NULL,
license_num      VARCHAR(40) NOT NULL,
order_id         INTEGER NOT NULL,
FOREIGN KEY (order_id) REFERENCES "order"(order_id));
```

COMMIT;

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer tree, which includes a 'Servers' node, a 'PostgreSQL 15' node, and a 'ITECH2004_30397109_DB_System' database node. Under the database node, there are 'Tables (4)' which include 'customer', 'order', 'payment', and 'promotion'. The main pane shows a SQL query editor with the following DDL code:

```

1 START TRANSACTION;
2
3 CREATE TABLE delivery_driver (
4     emp_id          INTEGER NOT NULL,
5     license_num     VARCHAR(40) NOT NULL,
6     order_id        INTEGER NOT NULL,
7     FOREIGN KEY (order_id) REFERENCES "order"(order_id);
8
9 COMMIT;

```

Below the query editor, the Data Output tab shows the message 'COMMIT' and 'Query returned successfully in 46 msec.' The status bar at the bottom indicates 'Total rows: 0 of 0' and 'Query complete 00:00:00.046'. A green success message box in the bottom right corner says 'Query returned successfully in 46 msec. Ln 7, Col 42'.

Figure 16 DDL for delivery_driver Table

3.13 manager

START TRANSACTION;

```

CREATE TABLE manager (
emp_id          INTEGER NOT NULL,
emp_fname        VARCHAR(40) NOT NULL,
emp_lname        VARCHAR(40) NOT NULL,
depart_name      VARCHAR(40) NOT NULL,
join_date        DATE DEFAULT CURRENT_TIMESTAMP NOT NULL);

```

COMMENT;

The screenshot shows the pgAdmin 15 interface. The left pane is the Object Explorer, displaying the database structure. The right pane is the Query Editor, showing the DDL code for creating a 'manager' table. The code is as follows:

```

1 START TRANSACTION;
2
3 CREATE TABLE manager (
4     emp_id      INTEGER NOT NULL,
5     depart_name VARCHAR(40) NOT NULL,
6     join_date   DATE DEFAULT CURRENT_TIMESTAMP NOT NULL);
7
8 COMMIT;

```

The 'Messages' tab in the Query Editor shows the message: 'Query returned successfully in 35 msec.' Below the editor, it says 'Total rows: 0 of 0' and 'Query complete 00:00:00.035'. A status bar at the bottom right indicates 'Ln 7, Col 1'.

Figure 17 DDL for manager Table

4. DML Statements

customer

cust_id	cust_fname	cust_lname	cust_address	cust_phone	cust_email
1001	Charlotte	Taylor	U16 80 Jacaranda rd Carringbah	04 5624-2093	taylorcharlotte@gmail.com
1002	Noah	Williams	U12 20 Judd rd Cronulla	04 7398-9023	williamsnoah@gmail.com
1003	Ava	Johnson	25 Rawson pde Carringbah	04 8383-7389	johnsonava@gmail.com
1004	Oliver	Jones	U26 40 Miranda rd Miranda	04 0139-2874	jonesoliver@gmail.com
1008	Ali	Mansour	U10 13-15 Sylvania rd Sylvania	04 1093-1083	mansourali@gmail.com
1005	Mia	Brown	15 Beach Rd Bondi Beach	04 7654-1234	miabrown@gmail.com
1006	Liam	Smith	8 Park Ave Randwick	04 8901-5678	liamsmith@gmail.com
1007	Olivia	Davis	6 Ocean St Coogee	04 6789-4321	oliviadavis@gmail.com
1009	Lucas	Lee	14 High St Maroubra	04 5555-5555	lucaslee@gmail.com
1010	Isabella	Hall	22 Sunset Blvd Bronte	04 2222-2222	isabellahall@gmail.com

START TRANSACTION;

```

/* customer rows */ 
INSERT INTO customer VALUES(1001,'Charlotte','Taylor','U16 80 Jacaranda rd Carringbah','04
5624-2093','taylorcharlotte@gmail.com');
INSERT INTO customer VALUES(1002,'Noah','Williams','U12 20 Judd rd Cronulla','04 7398-
9023','williamsnoah@gmail.com');
INSERT INTO customer VALUES(1003,'Ava','Johnson','25 Rawson pde Carringbah','04 8383-
7389','johnsonava@gmail.com');

```

```

INSERT INTO customer VALUES(1004,'Oliver','Jones','U26 40 Miranda rd Miranda','04 0139-2874','jonesoliver@gmail.com');
INSERT INTO customer VALUES(1008,'Ali', 'Mansour','U10 13-15 Sylvania rd Sylvania','04 1093-1083','mansourali@gmail.com');

COMMIT;

```

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database structure under "ITECH2004_30397109_DB_System".
- Query Editor:** Contains the following SQL code:


```

1 START TRANSACTION;
2
3 /* Loading data rows */
4
5 /* customer rows */
6 INSERT INTO customer VALUES(1001,'Charlotte','Taylor','U16 80 Jacaranda rd Canningbah','04 5624-21
7 INSERT INTO customer VALUES(1002,'Noah','Williams','U12 20 Judd rd Cronulla','04 7398-9023','will
8 INSERT INTO customer VALUES(1003,'Ava','Johnson','25 Rawson pde Canningbah','04 8383-7389','johns
9 INSERT INTO customer VALUES(1004,'Oliver','Jones','U26 40 Miranda rd Miranda','04 0139-2874','joni
10 INSERT INTO customer VALUES(1008,'Ali','Mansour','U10 13-15 Sylvania rd Sylvania','04 1093-1083'
11
12
13 COMMIT;
      
```
- Data Output:** Shows the message "Query returned successfully in 46 msec."
- Messages:** Shows the message "COMMIT".
- Notifications:** None.
- Status Bar:** Shows "Total rows: 0 of 0" and "Query complete 00:00:00.046".
- Bottom Status:** A green box indicates "Query returned successfully in 46 msec." with a close button.

Figure 18 DML for customer Table

promotion

promotion_id	promo_name	description	discount_percent	start_date	end_date
1	Summer Sale	Special discounts for the summer season	10	1/6/2023	31/8/2023
2	Holiday Promo	Promotion for the holiday season	15.5	15/11/2023	31/12/2023
3	Back to School	Discounts for back-to-school shopping	5	15/8/2023	30/9/2023
4	Spring Clearance	Clearance sale for spring	20	1/4/2023	15/5/2023
5	Black Friday	Black Friday deals	30	24/11/2023	26/11/2023
6	Winter Warm-up	Stay warm with winter discounts	12	1/12/2023	15/1/2024
7	Easter Sale	Easter-themed promotions	8	10/4/2023	16/4/2023
8	New Year Special	Welcome the new year with savings	10	1/1/2024	31/1/2024
9	Valentine's Day	Love-themed discounts	14	1/2/2024	14/2/2024
10	Spring Break	Promotions for spring break	15	1/3/2024	15/3/2024

START TRANSACTION;

```

/* promotion rows */
INSERT INTO promotion VALUES(1, 'Summer Sale', 'Special discounts for the summer
season', 10.00, '2023-06-01', '2023-08-31');
INSERT INTO promotion VALUES(2, 'Holiday Promo', 'Promotion for the holiday season',
15.50, '2023-11-15', '2023-12-31');
INSERT INTO promotion VALUES(3, 'Back to School', 'Discounts for back-to-school shopping',
5.00, '2023-08-15', '2023-09-30');
      
```

```

INSERT INTO promotion VALUES(4, 'Spring Clearance', 'Clearance sale for spring', 20.00,
'2023-04-01', '2023-05-15');
INSERT INTO promotion VALUES(5, 'Black Friday', 'Black Friday deals', 30.00, '2023-11-24',
'2023-11-26');

COMMIT;

```

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database structure with several tables: customer, order, payment, promotion, Trigger Functions, Types, Views, Subscriptions, and ITECH 2004_Enrollment.
- Query Editor:** Contains the following SQL code:


```

1 START TRANSACTION;
2
3 /* promotion rows */
4 INSERT INTO promotion VALUES(1, 'Summer Sale', 'Special discounts for the summer season', 10.00,
5 INSERT INTO promotion VALUES(2, 'Holiday Promo', 'Promotion for the holiday season', 15.50, '2023-
6 INSERT INTO promotion VALUES(3, 'Back to School', 'Discounts for back-to-school shopping', 5.00,
7 INSERT INTO promotion VALUES(4, 'Spring Clearance', 'Clearance sale for spring', 20.00, '2023-04-01',
8 INSERT INTO promotion VALUES(5, 'Black Friday', 'Black Friday deals', 30.00, '2023-11-24', '2023-11-26');
9
10 COMMIT;
      
```
- Data Output:** Shows the message "Query returned successfully in 194 msec."
- Messages:** Shows the message "COMMIT".
- Notifications:** No notifications are present.
- Status Bar:** Shows "Total rows: 0 of 0" and "Query complete 00:00:00.194".
- Bottom Right:** A green success message box: "✓ Query returned successfully in 194 msec. X Ln 8, Col 106".

Figure 19 DML for promotion Table

order

order_id	order_date	total_amount	cust_id	promotion_id
1	7/9/2023	50	1001	1
2	7/9/2023	75	1002	2
3	7/9/2023	30	1003	3
4	7/9/2023	40	1004	4
5	7/9/2023	60	1008	5
6	8/9/2023	55	1001	1
7	8/9/2023	70	1002	2
8	8/9/2023	35	1003	3
9	8/9/2023	45	1004	4
10	8/9/2023	65	1008	5

```
START TRANSACTION;
```

```

/* order rows */
INSERT INTO "order" VALUES(120, '2023-09-07', 50.00, 1001, 1);
INSERT INTO "order" VALUES(121, '2023-09-07', 75.00, 1002, 2);
INSERT INTO "order" VALUES(123, '2023-09-07', 30.00, 1003, 3);
INSERT INTO "order" VALUES(124, '2023-09-07', 40.00, 1004, 4);
INSERT INTO "order" VALUES(125, '2023-09-07', 60.00, 1008, 5);

```

```
COMMIT;
```

```

Object Explorer   Statistics  Dependencies  Dependents  lab 7/postgres...  ITECH2004_30397109_DB_System./postgres@PostgreSQL 15*
Servers (1)    PostgreSQL 15
  Databases (6)
    ITECH2004_30397109_DB_System.
      Casts
      Catalogs
      Event Triggers
      Extensions
      Foreign Data Wrappers
      Languages
      Publications
      Schemas (1)
        public
          Aggregates
          Collations
          Domains
          FTS Configurations
          FTS Dictionaries
          FTS Parsers
          FTS Templates
          Foreign Tables
          Functions
          Materialized Views
          Operators
          Procedures
          Sequences
        Tables (4)
          customer
          order
          payment
          promotion
        Trigger Functions
        Types
        Views
        Subscriptions
ITECH 2004_Enrollment

Query  Query History
1 START TRANSACTION;
2
3 /* order rows */
4 INSERT INTO "order" VALUES(120, '2023-09-07', 50.00, 1001, 1);
5 INSERT INTO "order" VALUES(121, '2023-09-07', 75.00, 1002, 2);
6 INSERT INTO "order" VALUES(123, '2023-09-07', 30.00, 1003, 3);
7 INSERT INTO "order" VALUES(124, '2023-09-07', 40.00, 1004, 4);
8 INSERT INTO "order" VALUES(125, '2023-09-07', 60.00, 1008, 5);
9
10 COMMIT;

Data Output  Messages  Notifications
COMMIT
Query returned successfully in 55 msec.

Total rows: 0 of 0  Query complete 00:00:00.055
  ✓ Query returned successfully in 55 msec. X
Ln 10, Col 8

```

Figure 20 DML for order Table

payment

payment_id	pay_date	amount	pay_method	order_id
11	8/9/2023	55	Credit Card	120
12	8/9/2023	70	PayPal	121
13	9/9/2023	45	Credit Card	123
15	9/9/2023	60	PayPal	124
16	10/9/2023	35	Credit Card	125
18	10/9/2023	80	PayPal	120
19	11/9/2023	42.5	Credit Card	121
20	11/9/2023	65	PayPal	123
21	12/9/2023	57	Credit Card	124
22	12/9/2023	75	PayPal	125

START TRANSACTION;

```

/* payment rows */ 
INSERT INTO payment VALUES(11, '2023-09-08', 55.00, 'Credit Card', 120);
INSERT INTO payment VALUES(12, '2023-09-08', 70.00, 'PayPal', 121);
INSERT INTO payment VALUES(13, '2023-09-09', 45.00, 'Credit Card', 123);
INSERT INTO payment VALUES(15, '2023-09-09', 60.00, 'PayPal', 124);
INSERT INTO payment VALUES(16, '2023-09-10', 35.00, 'Credit Card', 125);

```

COMMIT;

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying the database structure. The right pane is the Query Editor, showing a SQL script for inserting data into the payment table.

```

Object Explorer      Statistics  Dependencies  Dependents  lab 7/postgres...  ITECH2004_30397109_DB_System./postgres@PostgreSQL 15*
Servers (1)          |          |          |          |
  PostgreSQL 15     |          |          |          |
    Databases (6)   |          |          |          |
      ITECH2004_30397109_DB_System  |          |          |
        Casts          |          |          |
        Catalogs       |          |          |
        Event Triggers |          |          |
        Extensions     |          |          |
        Foreign Data Wrappers |          |          |
        Languages      |          |          |
        Publications   |          |          |
        Schemas (1)    |          |          |
          public        |          |          |
            Aggregates |          |          |
            Collations |          |          |
            Domains     |          |          |
            FTS Configurations |          |          |
            FTS Dictionaries |          |          |
            FTS Parsers   |          |          |
            FTS Templates |          |          |
            Foreign Tables |          |          |
            Functions    |          |          |
            Materialized Views |          |          |
            Operators    |          |          |
            Procedures   |          |          |
            Sequences    |          |          |
            Tables (4)   |          |          |
              customer  |          |          |
              order     |          |          |
              payment   |          |          |
              promotion |          |          |
            Trigger Functions |          |          |
            Types       |          |          |
            Views       |          |          |
            Subscriptions |          |          |
ITECH 2004_Enrollment |          |          |

```

Query History

```

1 START TRANSACTION;
2
3 /* payment rows */ /
4 INSERT INTO payment VALUES(11, '2023-09-08', 55.00, 'Credit Card', 120);
5 INSERT INTO payment VALUES(12, '2023-09-08', 78.00, 'PayPal', 121);
6 INSERT INTO payment VALUES(13, '2023-09-09', 45.00, 'Credit Card', 123);
7 INSERT INTO payment VALUES(15, '2023-09-09', 60.00, 'PayPal', 124);
8 INSERT INTO payment VALUES(16, '2023-09-10', 35.00, 'Credit Card', 125);
9
10 COMMIT;

```

Data Output

Messages

Notifications

COMMIT

Query returned successfully in 54 msec.

Total rows: 0 of 0 Query complete 00:00:00.054 Ln 10, Col 8

Figure 21 DML for payment Table

ingredient

ingredient_id	ing_name	ing_price
1001	Flour	5.99
1002	Tomatoes	1.99
1003	Cheese	3.49
1004	Pepperoni	4.99
1005	Mushrooms	2.49
1006	Onions	1.29
1007	Bell Peppers	1.79
1008	Sausage	4.49
1009	Olives	2.99
1010	Basil	1.49

START TRANSACTION;

```

/* ingredient rows */ /
INSERT INTO ingredient VALUES(1001, 'Flour', 5.99);
INSERT INTO ingredient VALUES(1002, 'Tomatoes', 1.99);
INSERT INTO ingredient VALUES(1003, 'Cheese', 3.49);
INSERT INTO ingredient VALUES(1004, 'Pepperoni', 4.99);
INSERT INTO ingredient VALUES(1005, 'Mushrooms', 2.49);

```

COMMIT;

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying the database structure. The right pane is the Query Editor, showing the SQL code for inserting data into the ingredient table. The status bar at the bottom indicates the query was completed successfully in 138 msec.

```

1 START TRANSACTION;
2
3 /* ingredient rows */
4 INSERT INTO ingredient VALUES(1001, 'Flour', 5.99);
5 INSERT INTO ingredient VALUES(1002, 'Tomatoes', 1.99);
6 INSERT INTO ingredient VALUES(1003, 'Cheese', 3.49);
7 INSERT INTO ingredient VALUES(1004, 'Pepperoni', 4.99);
8 INSERT INTO ingredient VALUES(1005, 'Mushrooms', 2.49);
9
10 COMMIT;

```

Total rows: 0 of 0 Query complete 00:00:00.138

✓ Query returned successfully in 138 msec. X
Ln 10, Col 8

Figure 22 DML for ingredient Table

topping

topping_id	top_name	top_price
2001	Pepperoni	1.99
2002	Mushrooms	1.49
2003	Onions	0.99
2004	Bell Peppers	1.29
2005	Sausage	2.49
2006	Olives	1.79
2007	Bacon	2.99
2008	Spinach	1.69
2009	Tomatoes	1.29
2010	Jalapeños	1.99

START TRANSACTION;

```

/* topping rows */ 
INSERT INTO topping VALUES(2003, 'Onions', 0.99);
INSERT INTO topping VALUES(2004, 'Bell Peppers', 1.29);
INSERT INTO topping VALUES(2005, 'Sausage', 2.49);
INSERT INTO topping VALUES(2008, 'Spinach', 1.69);
INSERT INTO topping VALUES(2010, 'Jalapeños', 1.99);

```

COMMIT;

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying the database structure. The right pane contains a query editor window with the following SQL code:

```

1 START TRANSACTION;
2
3 /* topping rows */
4 INSERT INTO topping VALUES(2003, 'Onions', 0.99);
5 INSERT INTO topping VALUES(2004, 'Bell Peppers', 1.29);
6 INSERT INTO topping VALUES(2005, 'Sausage', 2.49);
7 INSERT INTO topping VALUES(2008, 'Spinach', 1.69);
8 INSERT INTO topping VALUES(2010, 'Jalapeños', 1.99);
9
10 COMMIT;

```

The 'Messages' tab shows the message: "Query returned successfully in 56 msec." Below the messages, it says "Total rows: 0 of 0 Query complete 00:00:00.056". A status bar at the bottom right indicates "Ln 9, Col 1".

Figure 23 DML for topping Table

pizza

pizza_id	pizza_name	description	price	topping_id	ingredient_id	order_id
3001	Margherita	Classic margherita pizza	9.99	2003	1001	120
3002	Pepperoni	Delicious pepperoni pizza	11.99	2004	1004	121
3003	Vegetarian	Loaded with fresh veggies	10.99	2005	1002	123
3004	Meat Lovers	For the meat enthusiasts	12.99	2008	1003	124
3005	Supreme	A pizza with everything	13.99	2010	1005	125
3006	Hawaiian	Sweet and savory combo	11.99	2003	1002	120
3007	BBQ Chicken	Tangy BBQ sauce and chicken	12.99	2005	1003	121
3008	Mushroom Swiss	Rich and earthy flavors	10.99	2008	1004	123
3009	Veggie Delight	Packed with veggie goodness	10.99	2004	1005	124
3010	Four Cheese	A cheese lover's dream	11.99	2003	1001	125

START TRANSACTION;

```

/* pizza rows */
INSERT INTO pizza VALUES(3001, 'Margherita', 'Classic margherita pizza', 9.99, 2003, 1001, 120);
INSERT INTO pizza VALUES(3002, 'Pepperoni', 'Delicious pepperoni pizza', 11.99, 2004, 1004, 121);
INSERT INTO pizza VALUES(3003, 'Vegetarian', 'Loaded with fresh veggies', 10.99, 2005, 1002, 123);
INSERT INTO pizza VALUES(3004, 'Meat Lovers', 'For the meat enthusiasts', 12.99, 2008, 1003, 124);
INSERT INTO pizza VALUES(3005, 'Supreme', 'A pizza with everything', 13.99, 2010, 1005, 125);

```

COMMIT;

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying the database structure. The right pane is the Query Editor, showing the following SQL code:

```

1 START TRANSACTION;
2
3 /* pizza rows */ 
4 INSERT INTO pizza VALUES(3001, 'Margherita', 'Classic margherita pizza', 9.99, 2003, 1001, 128);
5 INSERT INTO pizza VALUES(3002, 'Pepperoni', 'Delicious pepperoni pizza', 11.99, 2004, 1004, 121);
6 INSERT INTO pizza VALUES(3003, 'Vegetarian', 'Loaded with fresh veggies', 10.99, 2005, 1002, 123);
7 INSERT INTO pizza VALUES(3004, 'Meat Lovers', 'For the meat enthusiasts!', 12.99, 2008, 1003, 124);
8 INSERT INTO pizza VALUES(3005, 'Supreme', 'A pizza with everything', 13.99, 2010, 1005, 125);
9
10 COMMIT;

```

The Messages tab shows the message "Query returned successfully in 38 msec." and the status bar indicates "Total rows: 0 of 0" and "Query complete 00:00:00.038".

Figure 24 DML for pizza Table

pizzatopping

pizzatopping_id	topping_id	pizza_id
4001	2003	3001
4002	2004	3002
4003	2005	3003
4004	2008	3004
4005	2010	3005
4006	2003	3006
4007	2004	3007
4008	2005	3008
4009	2008	3009
4010	2010	3010

START TRANSACTION;

```

/* pizzatopping rows */ 
INSERT INTO pizzatopping VALUES(4001, 2003, 3001);
INSERT INTO pizzatopping VALUES(4002, 2004, 3002);
INSERT INTO pizzatopping VALUES(4003, 2005, 3003);
INSERT INTO pizzatopping VALUES(4004, 2008, 3004);
INSERT INTO pizzatopping VALUES(4005, 2010, 3005);

```

COMMIT;

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying the database structure. The right pane is the Query Editor, showing a script of DML statements. The status bar at the bottom indicates the query was completed successfully in 58 msec.

```

1 START TRANSACTION;
2
3 /* pizzatopping rows */ 
4 INSERT INTO pizzatopping VALUES(4001, 2003, 3001);
5 INSERT INTO pizzatopping VALUES(4002, 2004, 3002);
6 INSERT INTO pizzatopping VALUES(4003, 2005, 3003);
7 INSERT INTO pizzatopping VALUES(4004, 2008, 3004);
8 INSERT INTO pizzatopping VALUES(4005, 2010, 3005);
9
10
11 COMMIT;

```

Total rows: 0 of 0 Query complete 00:00:00.058 ✓ Query returned successfully in 58 msec. Ln 10, Col 1

Figure 25 DML for pizzatopping Table

pizzaingredient

pizzaingredient_ingredient_id	pizza_id
5001	1001
5002	1004
5003	1002
5004	1003
5005	1005
5006	1001
5007	1003
5008	1004
5009	1002
5010	1005

START TRANSACTION;

```

/* pizzaingredient rows */ 
INSERT INTO pizzaingredient VALUES(5001, 1001, 3001);
INSERT INTO pizzaingredient VALUES(5002, 1004, 3001);
INSERT INTO pizzaingredient VALUES(5003, 1002, 3002);
INSERT INTO pizzaingredient VALUES(5004, 1003, 3002);
INSERT INTO pizzaingredient VALUES(5005, 1005, 3003);

```

COMMIT;

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying the database structure. The right pane is the Query Editor, showing the following SQL code:

```

1 START TRANSACTION;
2
3 /* pizzaingredient rows */ 
4 INSERT INTO pizzaingredient VALUES(5001, 1001, 3001);
5 INSERT INTO pizzaingredient VALUES(5002, 1004, 3001);
6 INSERT INTO pizzaingredient VALUES(5003, 1002, 3002);
7 INSERT INTO pizzaingredient VALUES(5004, 1003, 3002);
8 INSERT INTO pizzaingredient VALUES(5005, 1005, 3003);
9
10 COMMIT;

```

The Data Output tab shows the message "Query returned successfully in 57 msec." Below the tabs, it says "Total rows: 0 of 0 Query complete 00:00:00.057". A status bar at the bottom right indicates "Ln 8, Col 54".

Figure 26 DML for pizzaingredient Table

branch

branch_id	branch_name	location
6001	Downtown Pizza	123 Main Street, Downtown
6002	Crispy Crust Pizza	456 Elm Street, Uptown
6003	Pizzeria Italia	789 Oak Street, Little Italy
6004	Slice of Heaven	101 Pine Street, Midtown
6005	Pizza Palace	202 Maple Avenue, Central
6006	Manoosh Westside	303 Cedar Road, Westside
6007	ecManoosh East End	404 Birch Lane, East End
6008	Manoosh Northside	505 Walnut Avenue, Northside
6009	Manoosh Southside	606 Spruce Drive, Southside
6010	Manoosh Seaside	707 Fir Street, Seaside

START TRANSACTION;

```

/* branch rows */ 
INSERT INTO branch VALUES(6001, 'Downtown Pizza', '123 Main Street, Downtown');
INSERT INTO branch VALUES(6002, 'Crispy Crust Pizza', '456 Elm Street, Uptown');
INSERT INTO branch VALUES(6003, 'Pizzeria Italia', '789 Oak Street, Little Italy');
INSERT INTO branch VALUES(6004, 'Slice of Heaven', '101 Pine Street, Midtown');
INSERT INTO branch VALUES(6005, 'Pizza Palace', '202 Maple Avenue, Central');

```

COMMIT;

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying a tree structure of servers, databases, and schema objects. The right pane contains a query editor window titled 'ITECH2004_30397109_DB_System/postgres@PostgreSQL 15*'. The query is:

```

1 START TRANSACTION;
2
3 /* branch rows */
4 INSERT INTO branch VALUES(6001, 'Downtown Pizza', '123 Main Street, Downtown');
5 INSERT INTO branch VALUES(6002, 'Crispy Crust Pizza', '456 Elm Street, Uptown');
6 INSERT INTO branch VALUES(6003, 'Pizzeria Italia', '789 Oak Street, Little Italy');
7 INSERT INTO branch VALUES(6004, 'Slice of Heaven', '101 Pine Street, Midtown');
8 INSERT INTO branch VALUES(6005, 'Pizza Palace', '202 Maple Avenue, Central');
9
10
11 COMMIT;

```

The 'Messages' tab shows the message 'Query returned successfully in 55 msec.' at the bottom. A status bar at the bottom right indicates 'Ln 10, Col 1'.

Figure 27 DML for branch Table

employee

emp_id	emp_fname	emp_lname	position	salary	emp_phone	branch_id
7001	John	Smith	Manager	60000	3567890	6001
7002	Jane	Doe	Cashier	35000	45678901	6001
7003	Michael	Johnson	Cook	40000	46789012	6002
7004	Emily	Wilson	Waiter	32000	467890123	6002
7005	David	Brown	Manager	60000	57901234	6003
7006	Sarah	Lee	Cashier	35000	67812345	6003
7007	Robert	Davis	Cook	40000	78923456	6004
7008	Jennifer	Clark	Waiter	32000	81234567	6004
7009	William	Martinez	Manager	60000	12345678	6005
7010	Linda	Rodriguez	Cashier	35000	44567890	6005

START TRANSACTION;

```

/* employee rows */ 
INSERT INTO employee VALUES(7001, 'John', 'Smith', 'Manager', 60000.00, 3567890, 6001);
INSERT INTO employee VALUES(7002, 'Jane', 'Doe', 'Cashier', 35000.00, 45678901, 6001);
INSERT INTO employee VALUES(7003, 'Michael', 'Johnson', 'Cook', 40000.00, 46789012, 6002);
INSERT INTO employee VALUES(7004, 'Emily', 'Wilson', 'Waiter', 32000.00, 467890123, 6002);
INSERT INTO employee VALUES(7005, 'David', 'Brown', 'Manager', 60000.00, 57901234, 6003);

```

COMMIT;

```

Object Explorer   Statistics  Dependencies  Dependents  lab 7/postgres...  ITECH2004_30397109_DB_System./postgres@PostgreSQL 15*
Servers (1)    ITECH2004_30397109_DB_System
  PostgreSQL 15
    Databases (6)
      ITECH2004_30397109_DB_System
        Casts
        Catalogs
        Event Triggers
        Extensions
        Foreign Data Wrappers
        Languages
        Publications
        Schemas (1)
          public
            Aggregates
            Collations
            Domains
            FTS Configurations
            FTS Dictionaries
            FTS Parsers
            FTS Templates
            Foreign Tables
            Functions
            Materialized Views
            Operators
            Procedures
            Sequences
          Tables (4)
            customer
            order
            payment
            promotion
          Trigger Functions
          Types
          Views
          Subscriptions
        ITECH 2004_Enrollment
      Total rows: 0 of 0  Query complete 00:00:00.040
      Scratch Pad
      Query  Query History
      1 START TRANSACTION;
      2
      3 /* employee rows */ 
      4 INSERT INTO employee VALUES(7001, 'John', 'Smith', 'Manager', 60000.00, 3567890, 6001);
      5 INSERT INTO employee VALUES(7002, 'Jane', 'Doe', 'Cashier', 35000.00, 45678901, 6001);
      6 INSERT INTO employee VALUES(7003, 'Michael', 'Johnson', 'Cook', 40000.00, 46789012, 6002);
      7 INSERT INTO employee VALUES(7004, 'Emily', 'Wilson', 'Waiter', 32000.00, 467890123, 6002);
      8 INSERT INTO employee VALUES(7005, 'David', 'Brown', 'Manager', 60000.00, 57901234, 6003);
      9
      10
      11 COMMIT;
      COMMIT
      Query returned successfully in 40 msec.
      ✓ Query returned successfully in 40 msec. X
      Ln 11, Col 8
    
```

Figure 28 DML for employee Table

delivery_driver

emp_id	license_num	order_id
7016	DL12345	130
7017	DL23456	131
7018	DL34567	132
7019	DL45678	133
7020	DL56789	134
7021	DL67890	135
7022	DL78901	136
7023	DL89012	137
7024	DL90123	138
7025	DL01234	139

START TRANSACTION;

```

/* delivery_driver rows */ 
INSERT INTO delivery_driver VALUES(7006, 'DL12345', 120);
INSERT INTO delivery_driver VALUES(7007, 'DL23456', 121);
INSERT INTO delivery_driver VALUES(7008, 'DL34567', 123);
INSERT INTO delivery_driver VALUES(7009, 'DL45678', 124);
INSERT INTO delivery_driver VALUES(7010, 'DL45678', 125);
  
```

COMMIT;

```

Object Explorer   Statistics  Dependencies  Dependents  lab 7/postgres...  ITECH2004_30397109_DB_System./postgres@PostgreSQL 15*
Servers (1)    PostgreSQL 15
  Databases (6)
    ITECH2004_30397109_DB_System
      Casts
      Catalogs
      Event Triggers
      Extensions
      Foreign Data Wrappers
      Languages
      Publications
      Schemas (1)
        public
          Aggregates
          Collations
          Domains
          FTS Configurations
          FTS Dictionaries
          FTS Parsers
          FTS Templates
          Foreign Tables
          Functions
          Materialized Views
          Operators
          Procedures
          Sequences
        Tables (4)
          customer
          order
          payment
          promotion
        Trigger Functions
        Types
        Views
        Subscriptions
      ITECH 2004_Enrollment
ITECH2004_30397109_DB_System./postgres@PostgreSQL 15*
Query  Query History
1 START TRANSACTION;
2
3 /* delivery_driver rows */ 
4 INSERT INTO delivery_driver VALUES(7006, 'DL12345', 120);
5 INSERT INTO delivery_driver VALUES(7007, 'DL23456', 121);
6 INSERT INTO delivery_driver VALUES(7008, 'DL34567', 123);
7 INSERT INTO delivery_driver VALUES(7009, 'DL45678', 124);
8 INSERT INTO delivery_driver VALUES(7010, 'DL45678', 125);
9
10
11 COMMIT;

Data Output  Messages  Notifications
COMMIT
Query returned successfully in 61 msec.

Total rows: 0 of 0  Query complete 00:00:00.061
Ln 11, Col 8

```

Figure 29 DML for delivery_driver Table

manager

emp_id	emp_fname	emp_lname	depart_name	join_date
7026	Pasang	Sherpa	Operations	15/1/2020
7027	William	Davis	Marketing	20/3/2022
7028	Olivia	Wilson	Finance	5/11/2021
7029	Ethan	Brown	Sales	10/5/2020
7030	Sophia	Garcia	Operations	30/9/2023
7031	Mason	Martinez	IT	15/7/2022
7032	Amelia	Robinson	Customer Service	10/2/2022
7033	Liam	Clark	Production	3/12/2021
7034	Charlotte	Turner	Quality Control	20/8/2022
7035	James	Scott	Logistics	25/4/2021

START TRANSACTION;

```

/* manager rows */ 
INSERT INTO manager VALUES (7026, 'Pasang', 'Sherpa', 'Operations','2020-01-15');
INSERT INTO manager VALUES (7027, 'William', 'Davis', 'Marketing','2022-03-20');
INSERT INTO manager VALUES (7028, 'Olivia', 'Wilson', 'Finance','2021-11-05');
INSERT INTO manager VALUES (7029, 'Ethan', 'Brown', 'Sales','2020-05-10');
INSERT INTO manager VALUES (7030, 'Sophia', 'Garcia', 'Operations','2023-09-30');

```

COMMIT;

The screenshot shows the pgAdmin 4 interface. The left sidebar is the Object Explorer, displaying a tree view of database objects including Procedures, Sequences, Tables (13), Triggers, Functions, Types, Views, Subscriptions, and various schema and role entries. The 'promotion' table under 'Tables' is currently selected. The main pane contains a SQL query editor with the following code:

```

1 START TRANSACTION;
2
3 /* manager rows */
4 INSERT INTO manager VALUES (7026, 'Pasang', 'Sherpa', 'Operations', '2020-01-15');
5 INSERT INTO manager VALUES (7027, 'William', 'Davis', 'Marketing', '2022-03-20');
6 INSERT INTO manager VALUES (7028, 'Olivia', 'Wilson', 'Finance', '2021-11-05');
7 INSERT INTO manager VALUES (7029, 'Ethan', 'Brown', 'Sales', '2020-05-10');
8 INSERT INTO manager VALUES (7030, 'Sophia', 'Garcia', 'Operations', '2023-09-30');
9
10 COMMIT;
11

```

The 'Messages' tab in the bottom right shows the message: "Query returned successfully in 65 msec." Below the messages, it says "Total rows: 0 of 0 Query complete 00:00:00.065". A status bar at the bottom right indicates "Ln 11, Col 1".

Figure 30 DML for manager Table

5. Queries and Results

Query 1: Using INNER JOIN

START TRANSACTION;

```

SELECT
    "order".order_id,
    customer.cust_fname,
    customer.cust_lname,
    "order".order_date,
    "order".total_amount
FROM
    "order" AS "order"
INNER JOIN
    customer AS customer
ON
    "order".cust_id = customer.cust_id;

COMMIT;

```

```

Object Explorer   2004_30...  ITECH2004_30397109_DB_System./postgres@PostgreSQL 15*
Query  Execute/Refresh [F5]
1 START TRANSACTION;
2
3 SELECT
4     "order".order_id,
5     customer.cust_fname,
6     customer.cust_lname,
7     "order".order_date,
8     "order".total_amount
9 FROM
10    "order" AS "order"
11 INNER JOIN
12    customer AS customer
13 ON
14    "order".cust_id = customer.cust_id;
15
16 COMMIT;

Data Output  Messages  Notifications
Tables (13)  order_id  cust_fname  cust_lname  order_date  total_amount
              integer   character varying(35)  character varying(35)  date        numeric(8,2)

cust_id  cust_fname  cust_lname  order_date  total_amount
Total rows: 0 of 0  Query complete 00:00:00.047  ✓ Successfully run. Total query runtime: 47 msec. 0 rows affected.  Ln 14, Col 39

```

Figure 31 Query 1

Explanation:

By combining the "order" and "customer" columns using the shared cust_id column, this SQL query obtains a list of customer orders as well as customer information. In addition to the order ID, customer names, order dates, and total order amounts, it offers a thorough view of all customer orders.

Query 2: Using OUTER JOIN

START TRANSACTION;

```

SELECT
    customer.cust_id,
    customer.cust_fname,
    customer.cust_lname,
    customer.cust_address,
    customer.cust_phone,
    customer.cust_email,
    "order".order_id,
    "order".order_date,
    "order".total_amount
FROM
    customer
LEFT OUTER JOIN
    "order" ON customer.cust_id = "order".cust_id;

```

COMMIT;

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** On the left, it shows the database structure under "ITECH2004_30397109_DB_System". It includes sections for Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (1), public, Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (13).
- Query Editor:** The main area contains the following SQL code:

```

1 START TRANSACTION;
2
3 SELECT
4     customer.cust_id,
5     customer.cust_fname,
6     customer.cust_lname,
7     customer.cust_address,
8     customer.cust_phone,
9     customer.cust_email,
10    "order".order_id,
11    "order".order_date,
12    "order".total_amount
13 FROM
14     customer
15 LEFT OUTER JOIN
16     "order" ON customer.cust_id = "order".cust_id;
17
18 COMMIT;
19

```
- Data Output:** Below the query editor, there is a table header for the results:

cust_id	cust_fname	cust_lname	cust_address	cust_phone	cust_email	order_id	order_date	total_amount
---------	------------	------------	--------------	------------	------------	----------	------------	--------------
- Status Bar:** At the bottom, it shows "Total rows: 0 of 0" and "Query complete 00:00:00.044". A green message box indicates "Successfully run. Total query runtime: 44 msec. 0 rows affected.".

Figure 32 Query 2

Explanation:

Columns from the order and customer tables are being chosen. Even in cases when there are no orders (NULL values for order-related columns), we employ an LEFT OUTER JOIN to combine results from the customer table with matching rows from the order table.

Customers and any connected orders, including those who haven't placed orders, will be shown in this query.

Query 3: Using Specialization Hierarchy

START TRANSACTION;

```

SELECT
    emp_id,
    emp_fname,
    emp_lname,
    position
FROM
    employee
UNION ALL
SELECT
    emp_id,
    emp_fname,
    emp_lname,

```

```

depart_name AS position
FROM
manager;

```

COMMIT;

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer tree, which includes a 'Tables (13)' section containing 'promotion'. The main window displays a SQL query in the 'queryuptopromotionrow.sql*' tab:

```

1 START TRANSACTION;
2
3 SELECT
4     emp_id,
5     emp_fname,
6     emp_lname,
7     position
8 FROM
9     employee
10 UNION ALL
11 SELECT
12     emp_id,
13     emp_fname,
14     emp_lname,
15     depart_name AS position
16 FROM
17     manager;
18
19 COMMIT;
20

```

Below the query is a 'Data Output' pane showing the schema:

emp_id	emp_fname	emp_lname	position
--------	-----------	-----------	----------

At the bottom of the interface, a status bar indicates: 'Total rows: 0 of 0 Query complete 00:00:00.054' and 'Successfully run. Total query runtime: 54 msec. 0 rows affected.' with a green checkmark icon.

Figure 33 Query 3

Explanation:

The results of two SELECT queries are being combined using the UNION ALL operator. In the first query, information from the employee table is retrieved, including traits that apply to all employees. The second query obtains information from the manager table, which depicts an employee's area of specialisation. The same columns are returned by both queries: emp_id, emp_fname, emp_lname, and position. A comprehensive list of all employees, including managers, is provided by this query.

Query 4: Using GROUP BY and HAVING

START TRANSACTION;

```

SELECT
customer.cust_id,
cust_fname,
cust_lname,
SUM(total_amount) AS total_order_amount
FROM
customer
INNER JOIN
"order" ON customer.cust_id = "order".cust_id

```

GROUP BY

```
customer.cust_id,
cust_fname,
cust_lname
HAVING
SUM(total_amount) > 50;
```

COMMIT;

```
Object Explorer :CH2004_30... ITECH2004_30... ITECH2004_30... ITECH2004_30397109_DB_System./postgres@PostgreSQL 15*
Query History
1 START TRANSACTION;
2
3 SELECT
4     customer.cust_id,
5     cust_fname,
6     cust_lname,
7     SUM(total_amount) AS total_order_amount
8 FROM
9     customer
10    INNER JOIN
11        "order" ON customer.cust_id = "order".cust_id
12 GROUP BY
13    customer.cust_id,
14    cust_fname,
15    cust_lname
16 HAVING
17    SUM(total_amount) > 50;
18
19
```

cust_id	cust_fname	cust_lname	total_order_amount

Total rows: 0 of 0 Query complete 00:00:00.067 ✓ Successfully run. Total query runtime: 67 msec. 0 rows affected. Ln 13, Col 23

Figure 34 Query 4

Explanation:

Using the `cust_id` foreign key, we are linking the `customer` and `order` tables. In order to group the results by customers and determine the total order amount for each client, we utilise the `GROUP BY` command. The `HAVING` condition limits the results to clients with orders totalling more than \$50.

Query 5: Using INNER SQL Query

START TRANSACTION;

```
SELECT
cust_id,
cust_fname,
cust_lname,
cust_address,
```

```

    cust_phone,
    cust_email
FROM
    customer
WHERE
    cust_id IN (SELECT DISTINCT cust_id FROM "order");

COMMIT;

```

The screenshot shows the pgAdmin 4 interface. The left pane, titled 'Object Explorer', displays the database schema with various objects like Publications, Schemas, Tables, and Views. The 'customer' table is currently selected. The center pane is a 'Query' window containing the provided SQL code. The right pane shows the 'Data Output' tab, which includes a table definition for the 'customer' table with columns: cust_id, cust_fname, cust_lname, cust_address, cust_phone, and cust_email. Below the table definition, a message states 'Successfully run. Total query runtime: 49 msec. 0 rows affected.' and 'Ln 13, Col 55'.

cust_id	[PK] integer	cust_fname	character varying (35)	cust_lname	character varying (35)	cust_address	character varying (35)	cust_phone	character varying (12)	cust_email	character varying (35)

Figure 35 Query 5

Explanation:

From the customer table, we are choosing customer data. A list of unique customer IDs is retrieved from the order table using the INNER SQL statement (SELECT DISTINCT cust_id FROM "order"). Checking whether a customer's cust_id is on the list of customer IDs from the subquery is how the main query screens customers. This query returns information about customers who have made orders.

Query 6: Using UPDATE

```
START TRANSACTION;
```

```

UPDATE
    employee
SET
    position = 'Senior Cashier'
WHERE

```

```
position = 'Cashier';
```

```
COMMIT;
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** On the left, it shows the database schema with various objects like Publications, Schemas, Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables, branch, and customer.
- Query Editor:** The main area contains the following SQL code:


```

1 START TRANSACTION;
2
3 UPDATE
4   employee
5   SET
6     position = 'Senior Cashier'
7 WHERE
8   position = 'Cashier';
9
10 COMMIT;
11
      
```
- Messages Tab:** Below the editor, the "Messages" tab is selected, showing the message "Query returned successfully in 56 msec."
- Status Bar:** At the bottom, it displays "Total rows: 0 of 0" and "Query complete 00:00:00.056".
- Bottom Status:** A green status bar at the bottom right says "✓ Query returned successfully in 56 msec. X Ln 8, Col 26".

Figure 36 Query 6

Explanation:

The UPDATE statement is being used to change data in the employee table. For rows where the current position is "Cashier," we change the position column to "Senior Cashier." This query modifies the employee positions.

References

Zainurrohman, A. (2021, 04 18). *Subtypes and Supertypes*. Retrieved from medium.com:
<https://medium.com/nerd-for-tech/subtypes-and-supertypes-ef3b6b37c250>