Karachi, Pak

SALES OPTIMIZATION PLATFORM FOR PIZZERIA

Project Description:

The "Sales Optimization Platform for Pizzeria" is a revolutionary solution designed to elevate the sales and operational efficiency of pizzerias. This cutting-edge platform leverages advanced data analytics and interactive visualizations to transform raw data into actionable insights.

Abstract :

Our platform leverages a MySQL database for robust data management and Power BI for dynamic dashboard visualizations to analyze historical sales data, customer preferences, and market trends. This enables pizzerias to make strategic decisions that drive revenue growth, enhance customer satisfaction, and streamline operations. Key features include predicting peak sales periods, optimizing menus, and managing inventory efficiently to reduce waste and ensure top-selling items are always available. The user-friendly dashboards help pizzerias stay competitive, delight customers, and maximize profitability. Key outcomes include increased sales revenue, improved inventory management, enhanced customer insights, and greater operational efficiency.

Objectives :

- **Increase Sales Revenue**: Develop strategies to boost overall sales by identifying high-performing products and optimizing pricing.
- **Inventory Management**: Implement data-driven inventory control to minimize waste and ensure the availability of popular items.
- **Customer Insights**: Analyze customer behavior and preferences to tailor marketing campaigns and improve customer satisfaction.
- **Operational Efficiency**: Streamline operations by identifying bottlenecks and inefficiencies in the sales and delivery process.

Data Set Overview:

The dataset comprises several tables that encapsulate various aspects of the pizzeria's operations:

1. **Orders**: Records of all orders placed, including order ID, customer ID, order date, and total amount.
2. **Customers**: Information about customers, such as customer ID, name, contact details, and demographics.
3. **Items**: Details of menu items, including item ID, name, category, price, and ingredients.
4. **Addresses**: Addresses related to customers, including street, city, state, and postal code.
5. **Ingredients**: Information on ingredients used in menu items, including ingredient ID, name, and stock levels.
6. **Recipes**: Recipes for menu items, specifying the ingredients and quantities required for each item.
7. **Inventory**: Inventory levels of ingredients, including stock levels and reorder thresholds.
8. **Shifts**: Staff shift schedules, including shift ID, staff ID, date, start time, and end time.
9. **Staff**: Information about staff members, including staff ID, name, role, and contact details.
10. **Rota**: Staff rota schedules, detailing which staff members are assigned to which shifts.

BUILD DATABASE :

```
DROP DATABASE IF EXISTS my_pizza;
CREATE DATABASE my_pizza;
USE my_pizza;

CREATE TABLE orders (
    order_id int NOT NULL,
    pizza_id varchar(100) NOT NULL,
    quantity int NOT NULL,
    order_time time NOT NULL,
    Date date NOT NULL,
    pizza_size varchar(10) NOT NULL,
    pizza_category varchar(50) NOT NULL,
    pizza_name varchar(150) NOT NULL,
    Ordered_at varchar(100) NOT NULL,
    cust_address varchar(200) NOT NULL,
    Extra_Cheeze varchar(10) NOT NULL,
    Extra_Spicy varchar(10) NOT NULL,
    cust_id varchar(200) NOT NULL,
    item_id varchar(200) NOT NULL,
    add_id varchar(200) NOT NULL,
    recipe_id varchar(200) NOT NULL,
    ing_id varchar(200) NOT NULL,
    PRIMARY KEY (cust_id, add_id, item_id)
);

CREATE TABLE customers (
    cust_id varchar(200) NOT NULL,
    cust_firstname varchar(50) NOT NULL,
    cust_lastname varchar(50) NOT NULL,
    age int NOT NULL,
    total_price decimal(5,2) NOT NULL,
    city varchar(100) NOT NULL,
    gender varchar(10) NOT NULL,
    PRIMARY KEY (cust_id)
);
```

BUILD DATABASE :

```
CREATE TABLE address (
    add_id varchar(200) NOT NULL,
    delivery_address varchar(200) NOT NULL,
    delivery_city varchar(100) NOT NULL,
    state varchar(50) NOT NULL,
    postal_code int NOT NULL,
    latitude decimal(10,2) NOT NULL,
    longitude decimal(10,3) NOT NULL,
    PRIMARY KEY (add_id)
);

CREATE TABLE item (
    item_id varchar(200) NOT NULL,
    pizza_id varchar(100) NOT NULL,
    recipe_id varchar(200) NOT NULL,
    pizza_size varchar(10) NOT NULL,
    pizza_category varchar(50) NOT NULL,
    pizza_name varchar(150) NOT NULL,
    unit_price decimal(10,2) NOT NULL,
    total_price decimal(10,2) NOT NULL,
    PRIMARY KEY (item_id)
);

CREATE TABLE ingredient (
    ing_id varchar(200) NOT NULL,
    ing_name varchar(200) NOT NULL,
    ing_price decimal(8,2) NOT NULL,
    ing_weight int NOT NULL,
    ing_units varchar(50) NOT NULL,
    PRIMARY KEY (ing_id)
);
```

BUILD DATABASE :

```
CREATE TABLE recipe (
    row_id int NOT NULL,
    recipe_id varchar(200) NOT NULL,
    ing_id varchar(200) NOT NULL,
    quantity int NOT NULL,
    item_id varchar(200) NOT NULL,
    PRIMARY KEY (recipe_id, item_id)
);

CREATE TABLE inventory (
    inv_id varchar(200) NOT NULL,
    item_id varchar(200) NOT NULL,
    quantity int NOT NULL,
    PRIMARY KEY (inv_id)
);

CREATE TABLE shift (
    shift_id varchar(200) NOT NULL,
    day_of_week varchar(20) NOT NULL,
    start_time time NOT NULL,
    end_time time NOT NULL,
    PRIMARY KEY (shift_id)
);

CREATE TABLE staff (
    staff_id varchar(200) NOT NULL,
    first_name varchar(100) NOT NULL,
    last_name varchar(100) NOT NULL,
    hourly_hours int NOT NULL,
    position varchar(100) NOT NULL,
    MartialStatus varchar(50) NOT NULL,
    MonthlyIncome int NOT NULL,
    Over18 boolean NOT NULL,
    OverTime boolean NOT NULL,
    YearInCurrentRole int NOT NULL,
    PRIMARY KEY (staff_id)
);
```

BUILD DATABASE :

```
CREATE TABLE rota (
    row_id int NOT NULL,
    rota_id varchar(200) NOT NULL,
    Date date NOT NULL,
    shift_id varchar(200) NOT NULL,
    staff_id varchar(200) NOT NULL,
    PRIMARY KEY (row_id, shift_id, staff_id)
);

-- Foreign Key Constraints and Indexes

ALTER TABLE orders ADD CONSTRAINT fk_orders_cust_id FOREIGN KEY
(cust_id) REFERENCES customers (cust_id);
ALTER TABLE orders ADD CONSTRAINT fk_orders_add_id FOREIGN KEY
(add_id) REFERENCES address (add_id);
ALTER TABLE orders ADD CONSTRAINT fk_orders_item_id FOREIGN KEY
(item_id) REFERENCES item (item_id);

ALTER TABLE item ADD CONSTRAINT fk_item_recipe_id FOREIGN KEY
(recipe_id) REFERENCES recipe (recipe_id);

ALTER TABLE recipe ADD CONSTRAINT fk_recipe_ing_id FOREIGN KEY (ing_id)
REFERENCES ingredient (ing_id);

-- Step 1: Add an index on the item_id column in the recipe table
CREATE INDEX idx_recipe_item_id ON recipe(item_id);

-- Step 2: Add the foreign key constraint in the inventory table
ALTER TABLE inventory ADD CONSTRAINT fk_inventory_item_id FOREIGN KEY
(item_id) REFERENCES recipe (item_id);


ALTER TABLE rota ADD CONSTRAINT fk_rota_shift_id FOREIGN KEY (shift_id)
REFERENCES shift (shift_id);
ALTER TABLE rota ADD CONSTRAINT fk_rota_staff_id FOREIGN KEY (staff_id)
REFERENCES staff (staff_id);
```

BUILD DATABASE :

```
-- Indexes

CREATE INDEX idx_orders_add_id ON orders(add_id);
CREATE INDEX idx_orders_item_id ON orders(item_id);
CREATE INDEX idx_recipe_ing_id ON recipe(ing_id);
-- CREATE INDEX idx_recipe_item_id ON recipe(item_id);-
CREATE INDEX idx_rota_shift_id ON rota(shift_id);
-- CREATE INDEX idx_rota_staff_id ON rota(staff_id);
CREATE INDEX idx_orders_Date ON orders(Date);

-- Remove redundant index if exists
-- DROP INDEX IF EXISTS idx_rota_staff_id_new ON rota;

-- Ensure primary keys are unique and indexed
ALTER TABLE rota ADD CONSTRAINT fk_rota_Date FOREIGN KEY (Date)
REFERENCES orders (Date);
```

NORMALIZATION :

Pizza Optimization Platform For Pizzeria :

Normalization is a systematic approach to organizing a database to minimize redundancy and ensure data integrity. This report describes the normalization of the **Pizza Optimization Platform for Pizzeria** database up to the Third Normal Form (3NF) and includes details on foreign key constraints.

1. Orders Table

Column Name	*Data Type*	*Constraints*	*Description*
order_id	int	NOT NULL	Unique identifier for each order
pizza_id	varchar(100)	NOT NULL	Identifier for the pizza
quantity	int	NOT NULL	Number of pizzas ordered
order_time	time	NOT NULL	Time the order was placed
Date	date	NOT NULL, INDEX	Date the order was placed
pizza_size	varchar(10)	NOT NULL	Size of the pizza
pizza_category	varchar(50)	NOT NULL	Category of the pizza
pizza_name	varchar(150)	NOT NULL	Name of the pizza
Ordered_at	varchar(100)	NOT NULL	Location where order was placed
cust_address	varchar(200)	NOT NULL	Address of the customer
Extra_Cheeze	varchar(10)	NOT NULL	Whether extra cheese was added
Extra_Spicy	varchar(10)	NOT NULL	Whether extra spicy was requested
cust_id	varchar(200)	PRIMARY KEY, FOREIGN KEY REFERENCES customers(cust_id)	Customer identifier
item_id	varchar(200)	PRIMARY KEY, FOREIGN KEY REFERENCES item(item_id)	Item identifier
add_id	varchar(200)	PRIMARY KEY, FOREIGN KEY REFERENCES address(add_id)	Address identifier
recipe_id	varchar(200)	NOT NULL	Recipe identifier
ing_id	varchar(200)	NOT NULL	Ingredient identifier

Indexes:

- idx_orders_add_id on add_id
- idx_orders_item_id on item_id
- idx_orders_Date on Date

Normalization 1NF :

order_id	pizza_id	quantity	order_time	pizza_size	cust_address	cust_id	item_id	add_id	time_of_order	recipe_id	ing_id
1	hawaiian_m	1	11:38 AM	M	1200 N. Muldoon Road	CUS-001	IT-001	ADD-001	11:38:00	hawaiian_m	in-00 01
2	classic_dlx_m	1	11:57 AM	M	150 West 100th Ave	CUS-002	IT-002	ADD-002	11:57:00	hawaiian_m	in-00 02
3	five_cheese_l	1	11:57 AM	L	2220 Abbott Rd	CUS-003	IT-003	ADD-003	11:57:00	hawaiian_m	in-00 03
4	ital_supr_l	1	11:57 AM	L	341 Bonifac e Pkwy	CUS-004	IT-004	ADD-004	11:57:00	hawaiian_m	in-00 04
5	mexicana_m	1	11:57 AM	M	3727 Spenard Highway	CUS-005	IT-005	ADD-005	11:57:00	classic_dlx_m	in-00 05
6	thai_ckn_l	1	11:57 AM	L	11740 Old Glenn Hwy	CUS-006	IT-006	ADD-006	11:57:00	classic_dlx_m	in-00 06
7	ital_supr_m	1	12:12 PM	M	89 College Rd	CUS-007	IT-007	ADD-007	12:12:00	classic_dlx_m	in-00 07
8	prsc_argla_l	1	12:12 PM	L	257 N Santa Claus Ln	CUS-008	IT-008	ADD-008	12:12:00	classic_dlx_m	in-00 08

2. Customers Table

Column Name	*Data Type*	*Constraints*	*Description*
cust_id	varchar(200)	PRIMARY KEY	Unique identifier for customer
cust_firstname	varchar(50)	NOT NULL	Customer's first name
cust_lastname	varchar(50)	NOT NULL	Customer's last name
age	int	NOT NULL	Customer's age
total_price	decimal(5,2)	NOT NULL	Total price spent
city	varchar(100)	NOT NULL	Customer's city
gender	varchar(10)	NOT NULL	Customer's gender

Foreign Key:

- cust_id references orders(cust_id)

Normalization 2NF :

Customers_Details :

cust_id	age	city	cust_firstname	cust_lastname	gender
CUS-001	28	Anchorage	Edithe	Leggis	Female
CUS-002	21	Anchorage	Elwood	Catt	Male
CUS-003	20	Anchorage	Darby	Felgate	Male
CUS-004	66	Anchorage	Dominica	Pyle	Female
CUS-005	53	Anchorage	Bay	Pencost	Male
CUS-006	28	Eagle River	Lora	Durbann	Female
CUS-007	49	Fairbanks	Rand	Bram	Male
CUS-008	32	North Pole	Perceval	Dallosso	Male
CUS-009	69	Soldotna	Aleda	Pigram	Female

Orders_Details :

order_id	cust_id	total_price
1	CUS-001	13.25
2	CUS-002	16
3	CUS-003	18.5
4	CUS-004	20.75
5	CUS-005	16
6	CUS-006	20.75
7	CUS-007	16.5

---

3. Address Table

Column Name	*Data Type*	*Constraints*	*Description*
add_id	varchar(200)	PRIMARY KEY	Unique identifier for address
delivery_address	varchar(200)	NOT NULL	Delivery address
delivery_city	varchar(100)	NOT NULL	Delivery city
state	varchar(50)	NOT NULL	State
postal_code	int	NOT NULL	Postal code
latitude	decimal(10,2)	NOT NULL	Latitude coordinate
longitude	decimal(10,3)	NOT NULL	Longitude coordinate

Foreign Key:

- add_id references orders(add_id)

Normalization 3NF :

Delivery_Location :

add_id	delivery_address	city	state	postal_code	latitude	longitude
ADD-001	1200 N. Muldoon Road	Anchorage	AK	99504	61.2037	-149.745
ADD-002	150 West 100th Ave	Anchorage	AK	99515	61.1194	-149.897
ADD-003	2220 Abbott Rd	Anchorage	AK	99507	61.1535	-149.829
ADD-004	341 Boniface Pkwy	Anchorage	AK	99504	61.2037	-149.745
ADD-005	3727 Spenard Highway	Anchorage	AK	99517	61.1901	-149.936

ADD-006	11740 Old Glenn Hwy	Eagle River	AK	99577	61.3114	-149.509
ADD-007	89 College Rd	Fairbanks	AK	99701	64.8378	-147.716
ADD-008	257 N Santa Claus Ln	North Pole	AK	99705	64.7805	-147.369
ADD-009	44332 Sterling Hwy	Soldotna	AK	99669	60.4818	-151.136

Order_Addresses :

order_id	*add_id*
1	ADD-001
2	ADD-002
3	ADD-003
4	ADD-004
5	ADD-005
6	ADD-006
7	ADD-007

---

4. Item Table

Column Name	*Data Type*	*Constraints*	*Description*
item_id	varchar(200)	PRIMARY KEY	Unique identifier for item
pizza_id	varchar(100)	NOT NULL	Identifier for the pizza
recipe_id	varchar(200)	FOREIGN KEY REFERENCES recipe(recipe_id)	Identifier for the recipe
pizza_size	varchar(10)	NOT NULL	Size of the pizza
pizza_category	varchar(50)	NOT NULL	Category of the pizza
pizza_name	varchar(150)	NOT NULL	Name of the pizza
unit_price	decimal(10,2)	NOT NULL	Unit price of the pizza
total_price	decimal(10,2)	NOT NULL	Total price

Foreign Key:

- recipe_id references recipe(recipe_id)

Indexes:

- idx_orders_item_id on item_id

Reason For No Normalization:

 The table is in 2NF and already effectively separates the pizza attributes and their prices.

 It supports detailed item management and pricing without redundancy.

---

5. Ingredient Table

Column Name	Data Type	Constraints	Description
ing_id	varchar(200)	PRIMARY KEY	Unique identifier for ingredient
ing_name	varchar(200)	NOT NULL	Name of the ingredient
ing_price	decimal(8,2)	NOT NULL	Price of the ingredient
ing_weight	int	NOT NULL	Weight of the ingredient
ing_units	varchar(50)	NOT NULL	Units of the ingredient

Foreign Key:

- ing_id references recipe(ing_id)

Indexes:

- idx_recipe_ing_id on ing_id

Normalization 2NF

Ingredient_Catalog:

ing_id	ing_name	ing_price
in-0001	Pizza dough	7.5
in-0002	Tomato sauce	5
in-0003	Mozzarella cheese	12.5
in-0004	Pepperoni	15
in-0005	Mushrooms	7.5
in-0006	Bell peppers	5
in-0007	Onions	5
in-0008	Olives	7.5

Recipe_Composition

recipe_id	ing_id	item_id	quantity
hawaiian_m	in-0001	IT-001	250
hawaiian_m	in-0016	IT-002	80
hawaiian_m	in-0017	IT-003	170
hawaiian_m	in-0001	IT-004	250
classic_dlx_m	in-0019	IT-005	300
classic_dlx_m	in-0020	IT-006	100
classic_dlx_m	in-0001	IT-007	250

6. Recipe Table

Column Name	Data Type	Constraints	Description
row_id	int	NOT NULL	Unique row identifier
recipe_id	varchar(200)	PRIMARY KEY	Unique identifier for recipe
ing_id	varchar(200)	NOT NULL	Ingredient identifier
quantity	int	NOT NULL	Quantity of the ingredient used
item_id	varchar(200)	PRIMARY KEY	Item identifier

Foreign Key:

- ing_id references ingredient(ing_id)
- item_id references item(item_id)

Indexes:

idx_recipe_item_id on item_id.

---

7. Inventory Table

Column Name	Data Type	Constraints	Description
inv_id	varchar(200)	PRIMARY KEY	Unique identifier for inventory
item_id	varchar(200)	FOREIGN KEY REFERENCES recipe(item_id)	Item identifier
quantity	int	NOT NULL	Quantity in stock

Foreign Key:

- item_id references recipe(item_id)

Reason For No Normalization :

 The table is already normalized as it ensures that the quantity for each item is uniquely tracked by item_id.

 Further decomposition is unnecessary and could complicate the inventory management process.

---

8. Shift Table

Column Name	*Data Type*	*Constraints*	*Description*
shift_id	varchar(200)	PRIMARY KEY	Unique identifier for shift
day_of_week	varchar(20)	NOT NULL	Day of the week
start_time	time	NOT NULL	Start time of the shift
end_time	time	NOT NULL	End time of the shift

Foreign Key:

- shift_id references rota(shift_id)

Indexes:

- idx_rota_shift_id on shift_id

Reason For No Norma;ization :

 The attributes in this table (day of the week, start time, end time) are simple and there is no need to decompose them further.
 The current structure efficiently supports the requirement for scheduling shifts without causing data redundancy or anomalies..

---

9. Staff Table

Column Name	*Data Type*	*Constraints*	*Description*
staff_id	varchar(200)	PRIMARY KEY	Unique identifier for staff
first_name	varchar(100)	NOT NULL	Staff member's first name
last_name	varchar(100)	NOT NULL	Staff member's last name
hourly_hours	int	NOT NULL	Hours worked per hour
position	varchar(100)	NOT NULL	Position of the staff
MartialStatus	varchar(50)	NOT NULL	Marital status
MonthlyIncome	int	NOT NULL	Monthly income
Over18	boolean	NOT NULL	Is the staff over 18
OverTime	boolean	NOT NULL	Is overtime applicable
YearInCurrentRole	int	NOT NULL	Years in current role

Foreign Key:

- staff_id references rota(staff_id)

Indexes:

- idx_rota_staff_id_new on staff_id

Normalization:

 Each attribute in the Staff table is atomic and stores only a single piece of information. For example, first_name and last_name are not combined into a single attribute, ensuring data is stored in its simplest form.
 The table ensures that staff information is maintained without redundancy. Each staff member's details are unique and stored under a single identifier (staff_id), minimizing the risk of data anomalies.

 The attributes are dependent on the staff_id and not on each other, which satisfies the conditions for 2NF and 3NF. For example, first_name, last_name, position, and MonthlyIncome are all uniquely determined by the staff_id.

---

10. Rota Table

Column Name	*Data Type*	*Constraints*	*Description*
row_id	int	NOT NULL	Unique row identifier
rota_id	varchar(200)	NOT NULL	Unique identifier for rota
Date	date	FOREIGN KEY REFERENCES orders(Date)	Date of the rota
shift_id	varchar(200)	PRIMARY KEY, FOREIGN KEY REFERENCES shift(shift_id)	Shift identifier
staff_id	varchar(200)	PRIMARY KEY, FOREIGN KEY REFERENCES staff(staff_id)	Staff identifier

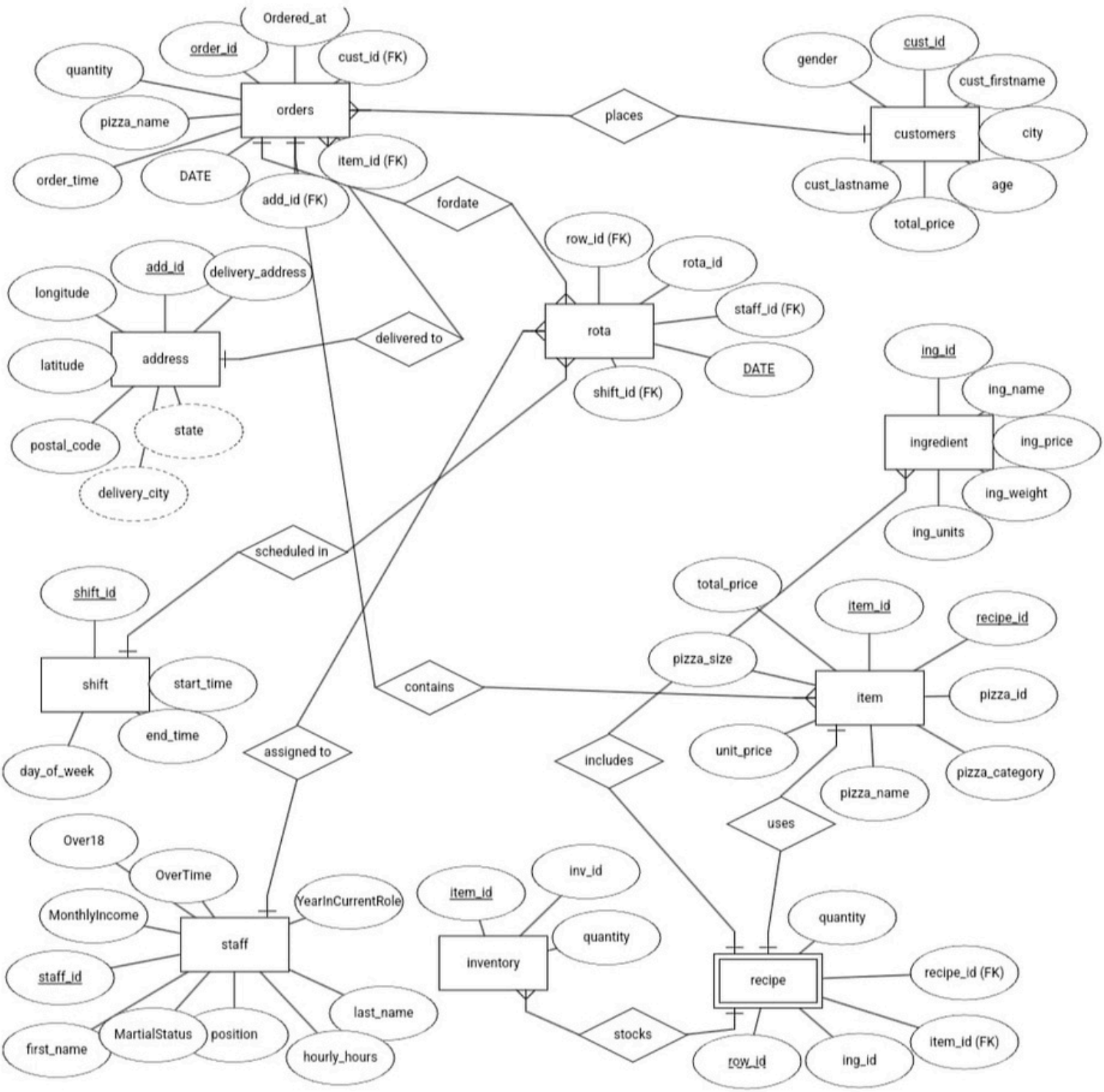
Indexes:

- idx_rota_staff_id on staff_id
- idx_rota_shift_id on shift_id

Reason For No Normalization :

 The table is in 1NF and is already serving its purpose effectively.

ER-DIAGRAM :


Ordered_at
order_id
cust_id (FK)
quantity
orders
pizza_name
order_time
DATE
item_id (FK)
add_id (FK)
fordate
places
gender
cust_id
cust_firstname
customers
city
cust_lastname
age
total_price
row_id (FK)
rota_id
staff_id (FK)
rota
DATE
shift_id (FK)
add_id
delivery_address
longitude
address
latitude
delivered to
postal_code
state
delivery_city
ing_id
ing_name
ingredient
ing_price
ing_weight
ing_units
scheduled in
shift_id
shift
start_time
end_time
day_of_week
total_price
item_id
recipe_id
pizza_size
item
pizza_id
contains
unit_price
pizza_category
pizza_name
includes
uses
assigned to
Over18
OverTime
MonthlyIncome
YearInCurrentRole
staff
staff_id
first_name
MartialStatus
position
hourly_hours
last_name
item_id
inv_id
quantity
inventory
stocks
quantity
recipe
recipe_id (FK)
item_id (FK)
row_id
ing_id

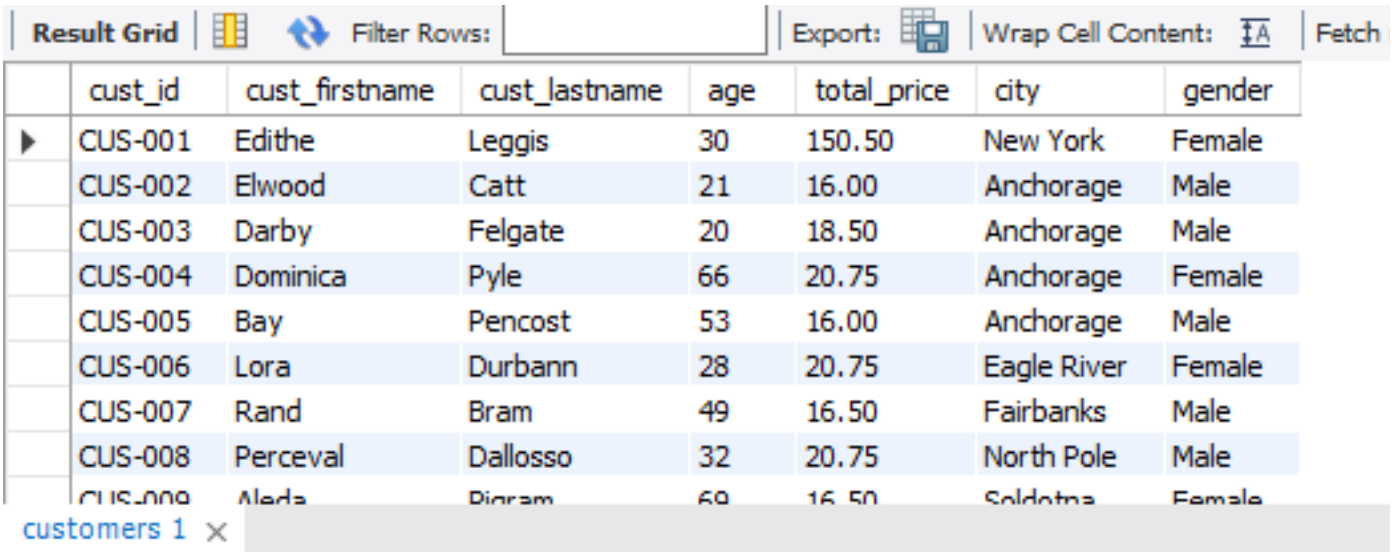
Data Definition Language (DDL) Queries:

1. Create Tables:

Query:

```
CREATE TABLE customers (
    cust_id varchar(200) NOT NULL,
    cust_firstname varchar(50) NOT NULL,
    cust_lastname varchar(50) NOT NULL,
    age int NOT NULL,
    total_price decimal(5,2) NOT NULL,
    city varchar(100) NOT NULL,
    gender varchar(10) NOT NULL,
    PRIMARY KEY (cust_id)
);
```

Output:



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

cust_id	cust_firstname	cust_lastname	age	total_price	city	gender
CUS-001	Edithe	Leggis	30	150.50	New York	Female
CUS-002	Elwood	Catt	21	16.00	Anchorage	Male
CUS-003	Darby	Felgate	20	18.50	Anchorage	Male
CUS-004	Dominica	Pyle	66	20.75	Anchorage	Female
CUS-005	Bay	Pencost	53	16.00	Anchorage	Male
CUS-006	Lora	Durbann	28	20.75	Eagle River	Female
CUS-007	Rand	Bram	49	16.50	Fairbanks	Male
CUS-008	Perceval	Dallosso	32	20.75	North Pole	Male
CUS-009	Aleda	Pigram	69	16.50	Soldotna	Female

customers 1

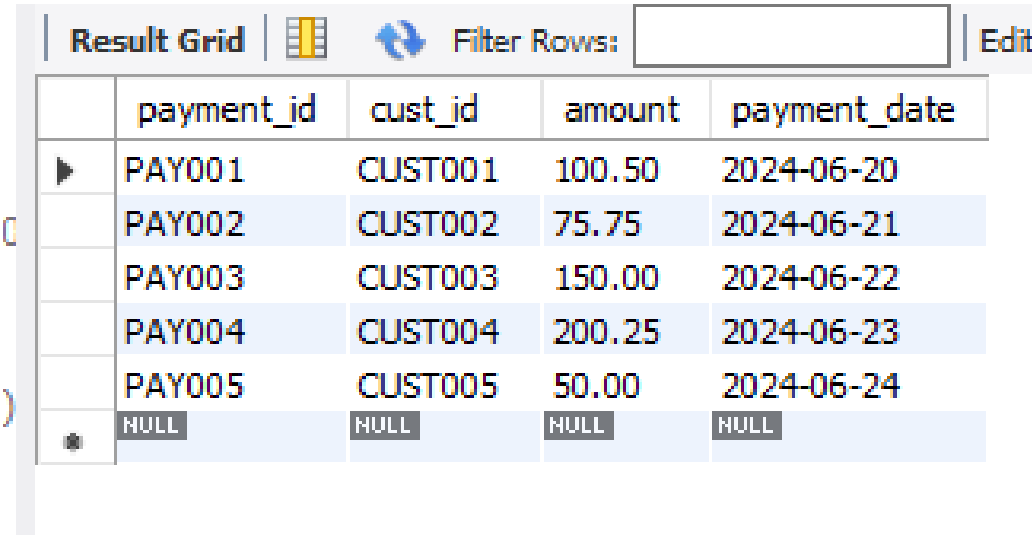
2. Alter Tables

Query:

```
-- Add a new column to the payment table
ALTER TABLE payments ADD COLUMN transaction_id varchar(200);
```

Output:

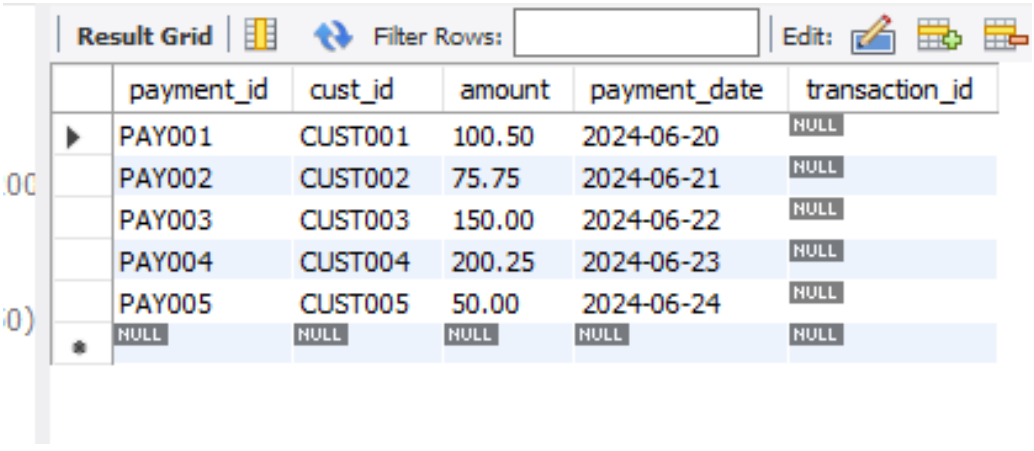
Before :



Result Grid | Filter Rows: | Edit

payment_id	cust_id	amount	payment_date
PAY001	CUST001	100.50	2024-06-20
PAY002	CUST002	75.75	2024-06-21
PAY003	CUST003	150.00	2024-06-22
PAY004	CUST004	200.25	2024-06-23
PAY005	CUST005	50.00	2024-06-24
NULL	NULL	NULL	NULL

After :



Result Grid | Filter Rows: | Edit:

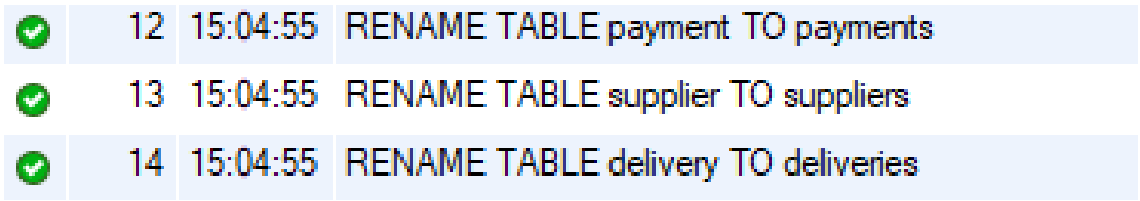
payment_id	cust_id	amount	payment_date	transaction_id
PAY001	CUST001	100.50	2024-06-20	NULL
PAY002	CUST002	75.75	2024-06-21	NULL
PAY003	CUST003	150.00	2024-06-22	NULL
PAY004	CUST004	200.25	2024-06-23	NULL
PAY005	CUST005	50.00	2024-06-24	NULL
NULL	NULL	NULL	NULL	NULL

3. Rename Tables:

Query:

```
-- Rename tables
RENAME TABLE payment TO payments;
RENAME TABLE supplier TO suppliers;
RENAME TABLE delivery TO deliveries;
```

Output:



✓	12	15:04:55	RENAME TABLE payment TO payments
✓	13	15:04:55	RENAME TABLE supplier TO suppliers
✓	14	15:04:55	RENAME TABLE delivery TO deliveries

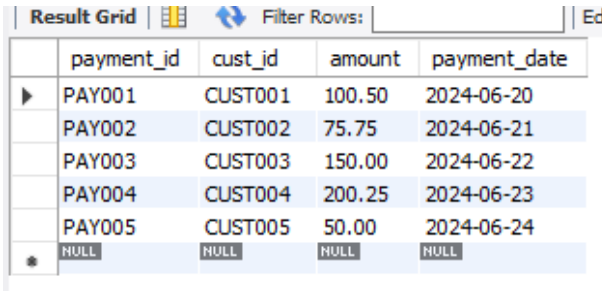
4. Truncate Tables:

Query:

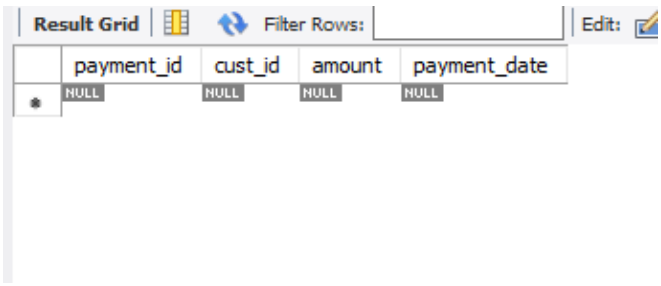
```
-- Truncate tables
TRUNCATE TABLE payments;
TRUNCATE TABLE suppliers;
TRUNCATE TABLE deliveries;
```

Output:

Payments Table:

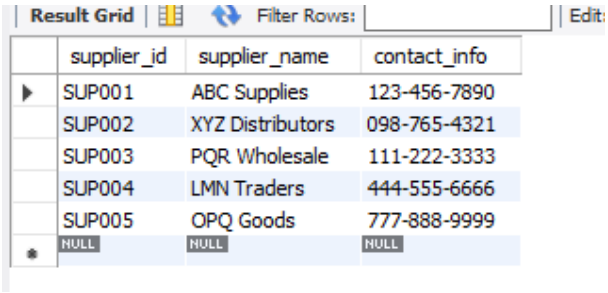


payment_id	cust_id	amount	payment_date
PAY001	CUST001	100.50	2024-06-20
PAY002	CUST002	75.75	2024-06-21
PAY003	CUST003	150.00	2024-06-22
PAY004	CUST004	200.25	2024-06-23
PAY005	CUST005	50.00	2024-06-24

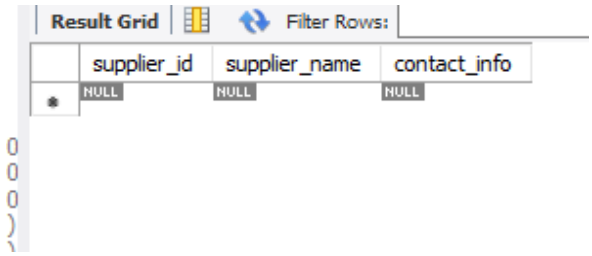


payment_id	cust_id	amount	payment_date
NULL	NULL	NULL	NULL

Suppliers Table:

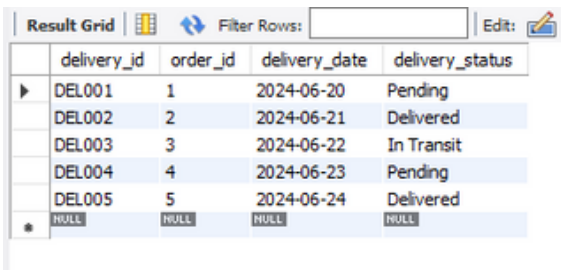


supplier_id	supplier_name	contact_info
SUP001	ABC Supplies	123-456-7890
SUP002	XYZ Distributors	098-765-4321
SUP003	PQR Wholesale	111-222-3333
SUP004	LMN Traders	444-555-6666
SUP005	OPQ Goods	777-888-9999

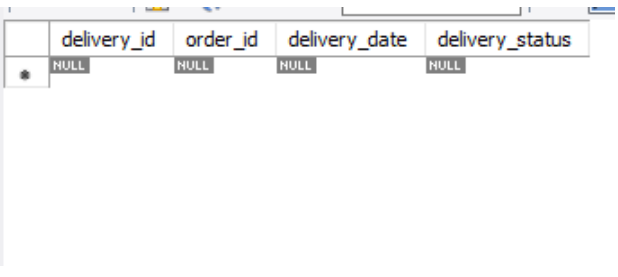


supplier_id	supplier_name	contact_info
NULL	NULL	NULL

Deliveries Table:



delivery_id	order_id	delivery_date	delivery_status
DEL001	1	2024-06-20	Pending
DEL002	2	2024-06-21	Delivered
DEL003	3	2024-06-22	In Transit
DEL004	4	2024-06-23	Pending
DEL005	5	2024-06-24	Delivered



delivery_id	order_id	delivery_date	delivery_status
NULL	NULL	NULL	NULL

5. Drop Tables:

Query:

```
-- Drop tables
DROP TABLE payments;
DROP TABLE suppliers;
DROP TABLE deliveries;
```

Output:

✓	39	15:18:51	DROP TABLE payments
✓	40	15:18:51	DROP TABLE suppliers
✓	41	15:18:51	DROP TABLE deliveries

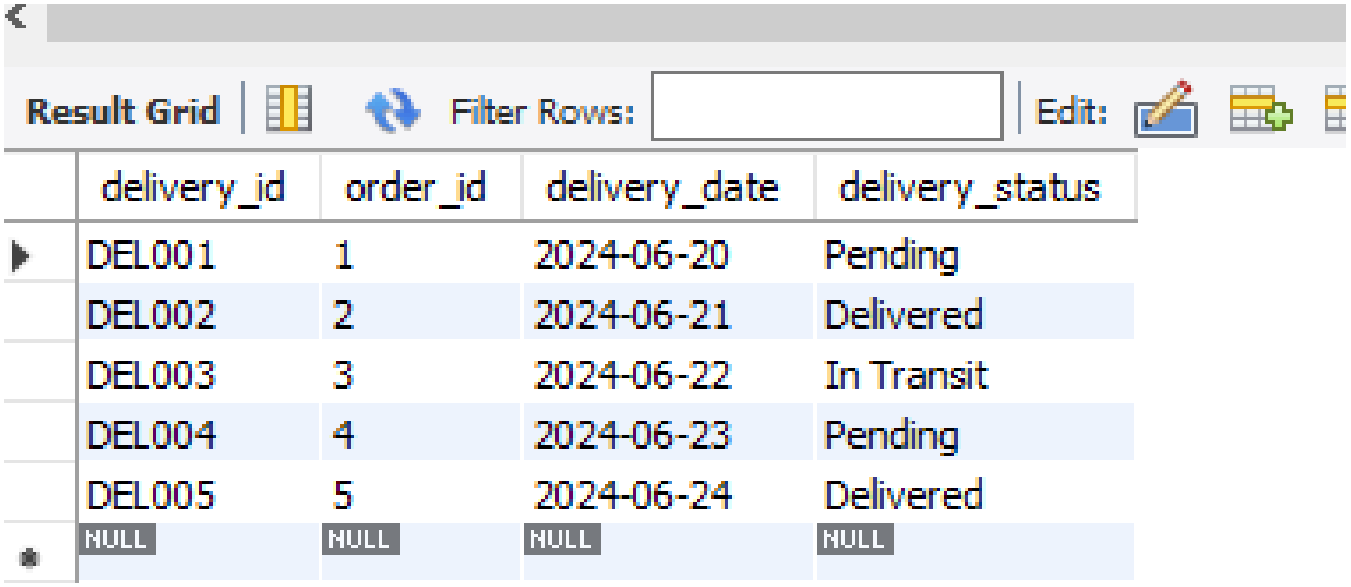
Data Manipulation Language (DML) Queries:

1. Insert Data Into Tables:

Query:

```
INSERT INTO deliveries (delivery_id, order_id, delivery_date,
delivery_status)
VALUES
('DEL001', 1, '2024-06-20', 'Pending'),
('DEL002', 2, '2024-06-21', 'Delivered'),
('DEL003', 3, '2024-06-22', 'In Transit'),
('DEL004', 4, '2024-06-23', 'Pending'),
('DEL005', 5, '2024-06-24', 'Delivered');
```

Output:



delivery_id	order_id	delivery_date	delivery_status
DEL001	1	2024-06-20	Pending
DEL002	2	2024-06-21	Delivered
DEL003	3	2024-06-22	In Transit
DEL004	4	2024-06-23	Pending
DEL005	5	2024-06-24	Delivered
NULL	NULL	NULL	NULL

2.Update Data in Tables:

Query:

```
UPDATE customers
SET
    city = 'New York',
    total_price = 150.50,
    age = 30
WHERE cust_id = 'CUS-001';
```

Output:

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

cust_id	cust_firstname	cust_lastname	age	total_price	city	gender
CUS-001	Edithe	Leggis	30	150.50	New York	Female

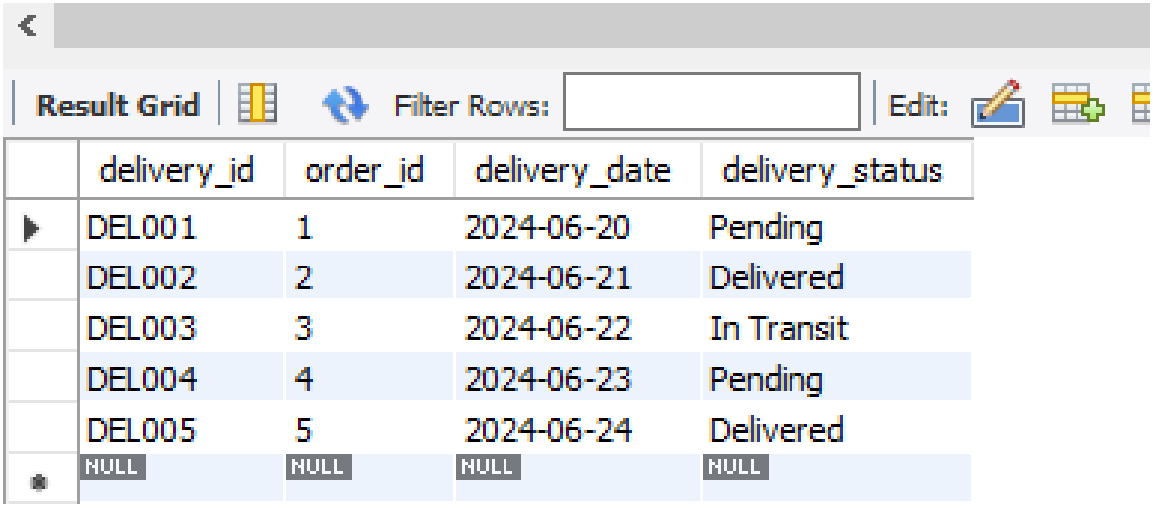
3. Delete Data from Tables:

Query:

```
DELETE FROM deliveries WHERE delivery_id = 'DEL001';
```

Output:

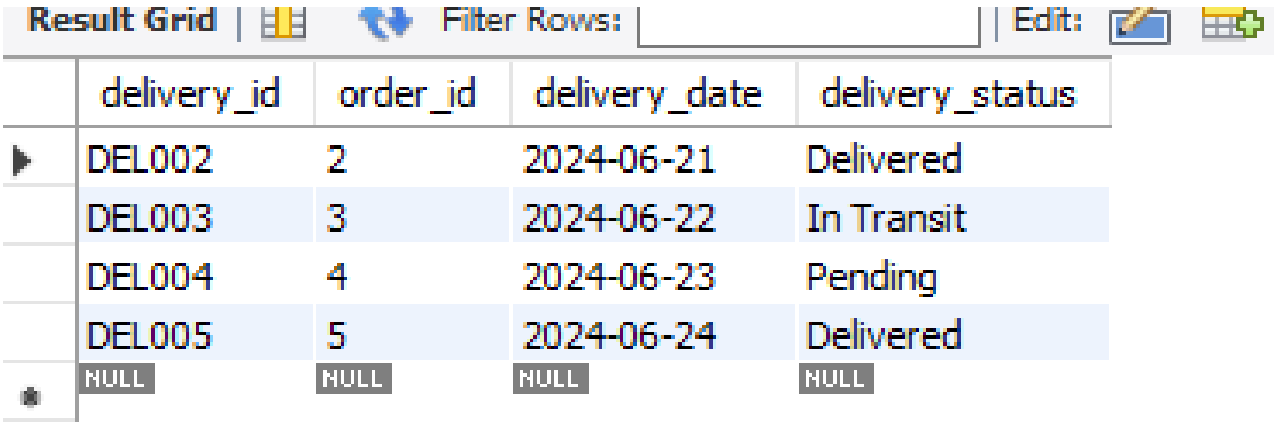
Before :



Result Grid | Filter Rows: | Edit:

delivery_id	order_id	delivery_date	delivery_status
DEL001	1	2024-06-20	Pending
DEL002	2	2024-06-21	Delivered
DEL003	3	2024-06-22	In Transit
DEL004	4	2024-06-23	Pending
DEL005	5	2024-06-24	Delivered
NULL	NULL	NULL	NULL

After :



Result Grid | Filter Rows: | Edit:

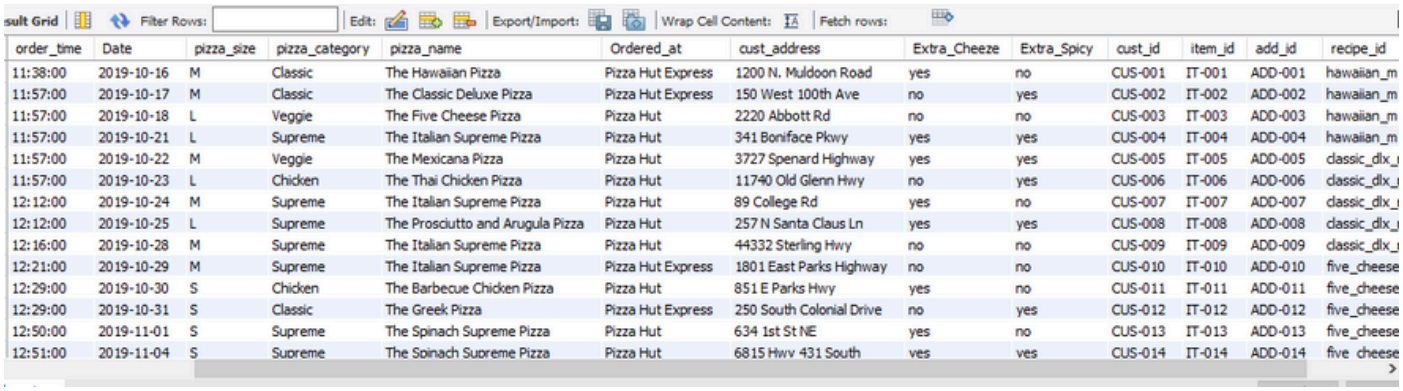
delivery_id	order_id	delivery_date	delivery_status
DEL002	2	2024-06-21	Delivered
DEL003	3	2024-06-22	In Transit
DEL004	4	2024-06-23	Pending
DEL005	5	2024-06-24	Delivered
NULL	NULL	NULL	NULL

4. Select Data from Tables:

-Query:

```
-- Select all data from orders
SELECT * FROM orders;
```

Output:

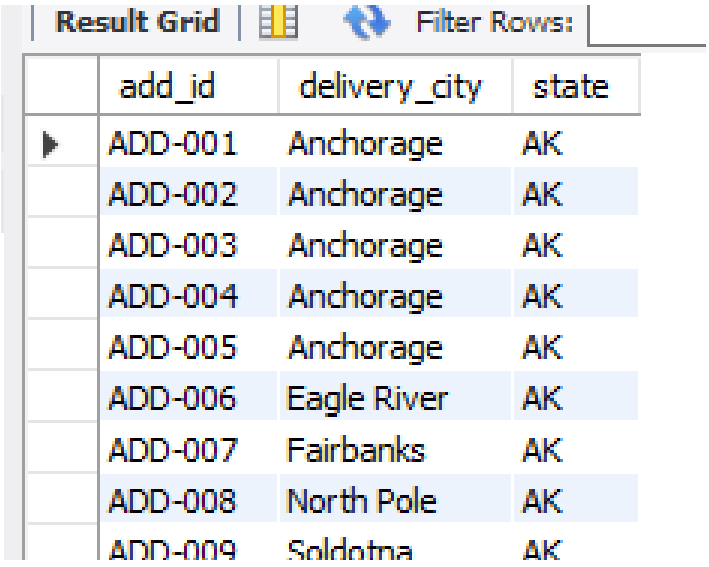


order_time	Date	pizza_size	pizza_category	pizza_name	Ordered_at	cust_address	Extra_Cheeze	Extra_Spicy	cust_id	item_id	add_id	recipe_id
11:38:00	2019-10-16	M	Classic	The Hawaiian Pizza	Pizza Hut Express	1200 N. Muldoon Road	yes	no	CUS-001	IT-001	ADD-001	hawaiian_m
11:57:00	2019-10-17	M	Classic	The Classic Deluxe Pizza	Pizza Hut Express	150 West 100th Ave	no	yes	CUS-002	IT-002	ADD-002	hawaiian_m
11:57:00	2019-10-18	L	Veggie	The Five Cheese Pizza	Pizza Hut	2220 Abbott Rd	no	no	CUS-003	IT-003	ADD-003	hawaiian_m
11:57:00	2019-10-21	L	Supreme	The Italian Supreme Pizza	Pizza Hut	341 Boniface Pkwy	yes	yes	CUS-004	IT-004	ADD-004	hawaiian_m
11:57:00	2019-10-22	M	Veggie	The Mexicana Pizza	Pizza Hut	3727 Spenard Highway	yes	yes	CUS-005	IT-005	ADD-005	classic_dlx_
11:57:00	2019-10-23	L	Chicken	The Thai Chicken Pizza	Pizza Hut	11740 Old Glenn Hwy	no	yes	CUS-006	IT-006	ADD-006	classic_dlx_
12:12:00	2019-10-24	M	Supreme	The Italian Supreme Pizza	Pizza Hut	89 College Rd	yes	no	CUS-007	IT-007	ADD-007	classic_dlx_
12:12:00	2019-10-25	L	Supreme	The Prosciutto and Arugula Pizza	Pizza Hut	257 N Santa Claus Ln	yes	yes	CUS-008	IT-008	ADD-008	classic_dlx_
12:16:00	2019-10-28	M	Supreme	The Italian Supreme Pizza	Pizza Hut	44332 Sterling Hwy	no	no	CUS-009	IT-009	ADD-009	classic_dlx_
12:21:00	2019-10-29	M	Supreme	The Italian Supreme Pizza	Pizza Hut Express	1801 East Parks Highway	no	no	CUS-010	IT-010	ADD-010	five_cheese
12:29:00	2019-10-30	S	Chicken	The Barbecue Chicken Pizza	Pizza Hut	851 E Parks Hwy	yes	no	CUS-011	IT-011	ADD-011	five_cheese
12:29:00	2019-10-31	S	Classic	The Greek Pizza	Pizza Hut Express	250 South Colonial Drive	no	yes	CUS-012	IT-012	ADD-012	five_cheese
12:50:00	2019-11-01	S	Supreme	The Spinach Supreme Pizza	Pizza Hut	634 1st St NE	yes	no	CUS-013	IT-013	ADD-013	five_cheese
12:51:00	2019-11-04	S	Supreme	The Spinach Supreme Pizza	Pizza Hut	6815 Hwy 431 South	yes	yes	CUS-014	IT-014	ADD-014	five_cheese

-Query:

```
-- Select specific columns from address
SELECT add_id, delivery_city, state FROM address;
```

Output:



Result Grid | Filter Rows:

add_id	delivery_city	state
ADD-001	Anchorage	AK
ADD-002	Anchorage	AK
ADD-003	Anchorage	AK
ADD-004	Anchorage	AK
ADD-005	Anchorage	AK
ADD-006	Eagle River	AK
ADD-007	Fairbanks	AK
ADD-008	North Pole	AK
ADD-009	Soldotna	AK

Join Queries:

1. Join Orders with Items and Addresses

Query:

```
SELECT
    o.order_id,
    i.total_price,
    o.quantity,
    i.pizza_category,
    i.pizza_name,
    o.order_time,
    a.delivery_address,
    a.state,
    a.delivery_city,
    a.postal_code,
    a.latitude,
    a.longitude
FROM orders o
LEFT JOIN item i ON o.item_id = i.item_id
LEFT JOIN address a ON o.add_id = a.add_id;
```

Output:

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

order_id	total_price	quantity	pizza_category	pizza_name	order_time	delivery_address	state	delivery_city	postal_code	latitude	longitude
1	13.25	3	Classic	The Hawaiian Pizza	11:38:00	1200 N. Muldoon Road	AK	Anchorage	99504	61.20	-149.745
2	16.00	1	Classic	The Classic Deluxe Pizza	11:57:00	150 West 100th Ave	AK	Anchorage	99515	61.12	-149.897
3	18.50	1	Veggie	The Five Cheese Pizza	11:57:00	2220 Abbott Rd	AK	Anchorage	99507	61.15	-149.829
4	20.75	1	Supreme	The Italian Supreme Pizza	11:57:00	341 Boniface Pkwy	AK	Anchorage	99504	61.20	-149.745
5	16.00	1	Veggie	The Mexicana Pizza	11:57:00	3727 Spenard Highway	AK	Anchorage	99517	61.19	-149.936
6	20.75	1	Chicken	The Thai Chicken Pizza	11:57:00	11740 Old Glenn Hwy	AK	Eagle River	99577	61.31	-149.509
7	16.50	1	Supreme	The Italian Supreme Pizza	12:12:00	89 College Rd	AK	Fairbanks	99701	64.84	-147.716
8	20.75	1	Supreme	The Prosciutto and Arugula Pizza	12:12:00	257 N Santa Claus Ln	AK	North Pole	99705	64.78	-147.369
9	16.50	1	Supreme	The Italian Supreme Pizza	12:16:00	44332 Sterling Hwy	AK	Soldotna	99669	60.48	-151.136
10	16.50	1	Supreme	The Italian Supreme Pizza	12:21:00	1801 East Parks Highway	AK	Wasilla	99654	61.59	-149.396

Result 1

2. Total Orders of Different Pizzas:

Query:

```
-- Total orders of different pizzas
SELECT
    i.pizza_name,
    COUNT(*) AS total_orders
FROM
    orders o
JOIN
    item i ON o.item_id = i.item_id
GROUP BY
    i.pizza_name
ORDER BY
    total_orders DESC;
```

Output:

Result Grid | Filter Rows:

	pizza_category	total_revenue
▶	Classic	4728.50
	Supreme	4254.95
	Chicken	4183.50
	Veggie	4074.35

3. Calculate Profit Margin for Each Pizza Size:

Query:

```
--  Calculate the Profit Margin for Each Pizza Size
 SELECT
    i.pizza_size,
    SUM(o.quantity * i.unit_price) AS total_revenue,
    SUM(o.quantity * i.unit_price) AS total_profit,
    1.0 AS profit_margin
FROM
    orders o
JOIN
    item i ON o.item_id = i.item_id
GROUP BY
    i.pizza_size
ORDER BY
    profit_margin DESC;
```

Output:

Result Grid | Filter Rows: | Export:

pizza_size	total_revenue	total_profit	profit_margin
M	5368.50	5368.50	1.0
L	8087.35	8087.35	1.0
S	3530.45	3530.45	1.0
XL	255.00	255.00	1.0

4. Staff and Shift Details:

Query:

```
select r.Date,
 s.first_name,
 s.last_name,
 s.hourly_hours  ,
 sh.start_time,
 sh.end_time,
round(((hour(timediff(sh.end_time,sh.start_time)) * 60 ) +
(minute(timediff(sh.end_time,sh.start_time)))) / 60,2)as hours_in_shift,
round(((hour(timediff(sh.end_time,sh.start_time)) * 60 ) +
(minute(timediff(sh.end_time,sh.start_time)))) / 60 * s.hourly_hours,2) as staff_cost
from rota as r
left join staff as s
on r.staff_id = s.staff_id
left join shift as sh
on r.shift_id = sh.shift_id
where s.staff_id is not NULL;
```

Output:

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

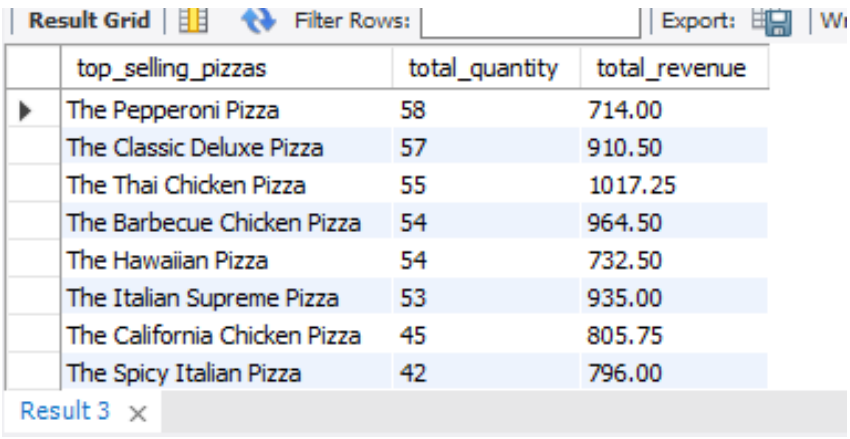
Date	first_name	last_name	hourly_hours	start_time	end_time	hours_in_shift	staff_cost
2019-10-16	Lavine	Bianchi	12	14:30:00	18:00:00	3.50	42.00
2019-10-17	Hargrave	Tyler	10	10:30:00	14:00:00	3.50	35.00
2019-10-18	Hill	Martin	11	11:30:00	15:00:00	3.50	38.50
2019-10-21	Onio	Okagbue	8	12:30:00	16:00:00	3.50	28.00
2019-10-22	Boni	Yin	7	13:30:00	17:00:00	3.50	24.50
2019-10-23	Mitchell	Buccho	14	14:30:00	18:00:00	3.50	49.00
2019-10-24	Chu	Chidiebele	14	15:30:00	19:00:00	3.50	49.00
2019-10-25	Bartlett	Trevisani	13	16:30:00	20:00:00	3.50	45.50

5. Find the Top-Selling Pizzas by Quantity and Revenue with Pizza Details:

Query:

```
SELECT
    i.pizza_name as top_selling_pizzas,
    SUM(o.quantity) AS total_quantity,
    SUM(o.quantity * i.unit_price) AS total_revenue
FROM
    orders o
JOIN
    item i ON o.item_id = i.item_id
GROUP BY
    i.pizza_name
ORDER BY
    total_quantity DESC, total_revenue DESC;
```

Output:


Result Grid | Filter Rows: | Export: | W

top_selling_pizzas	total_quantity	total_revenue
The Pepperoni Pizza	58	714.00
The Classic Deluxe Pizza	57	910.50
The Thai Chicken Pizza	55	1017.25
The Barbecue Chicken Pizza	54	964.50
The Hawaiian Pizza	54	732.50
The Italian Supreme Pizza	53	935.00
The California Chicken Pizza	45	805.75
The Spicy Italian Pizza	42	796.00

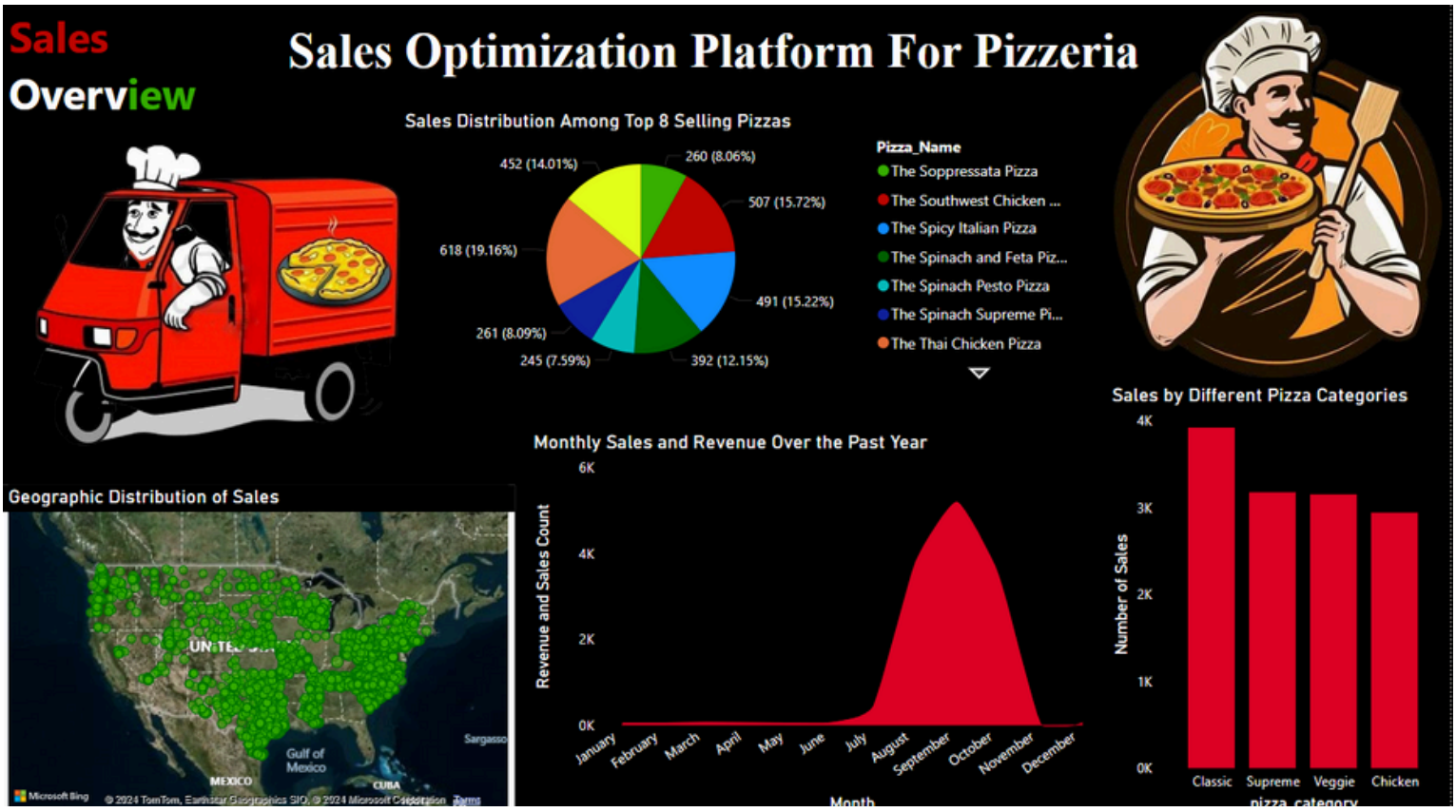
Result 3

DASHBOARD :

Our dashboard comprises of 4 pages :

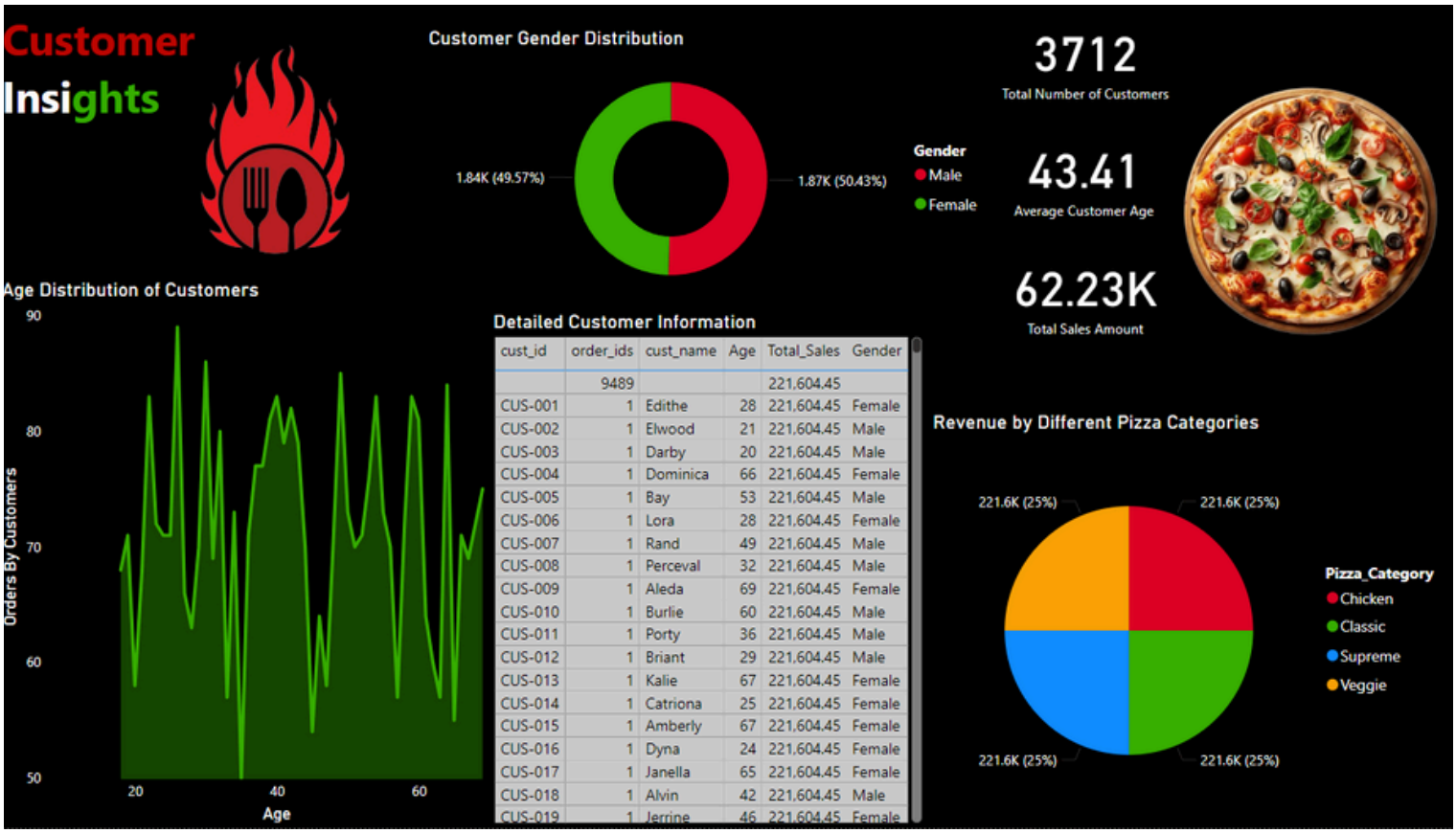
1. Sales Overview:

Insights: The "Sales Overview" section of this dashboard offers a concise and insightful summary of the pizzeria's sales performance. It leverages various visualizations, including the **Sales Distribution Among Top 8 Selling Pizzas** pie chart, the **Geographic Distribution of Sales** map, the **Monthly Sales and Revenue Over the Past Year** area chart, and the **Sales by Different Pizza Categories** bar chart. These visualizations highlight key metrics and trends, providing a clear understanding of sales activities. This section is designed to help identify top-selling items, sales distributions, and regional preferences, enabling data-driven decision-making.



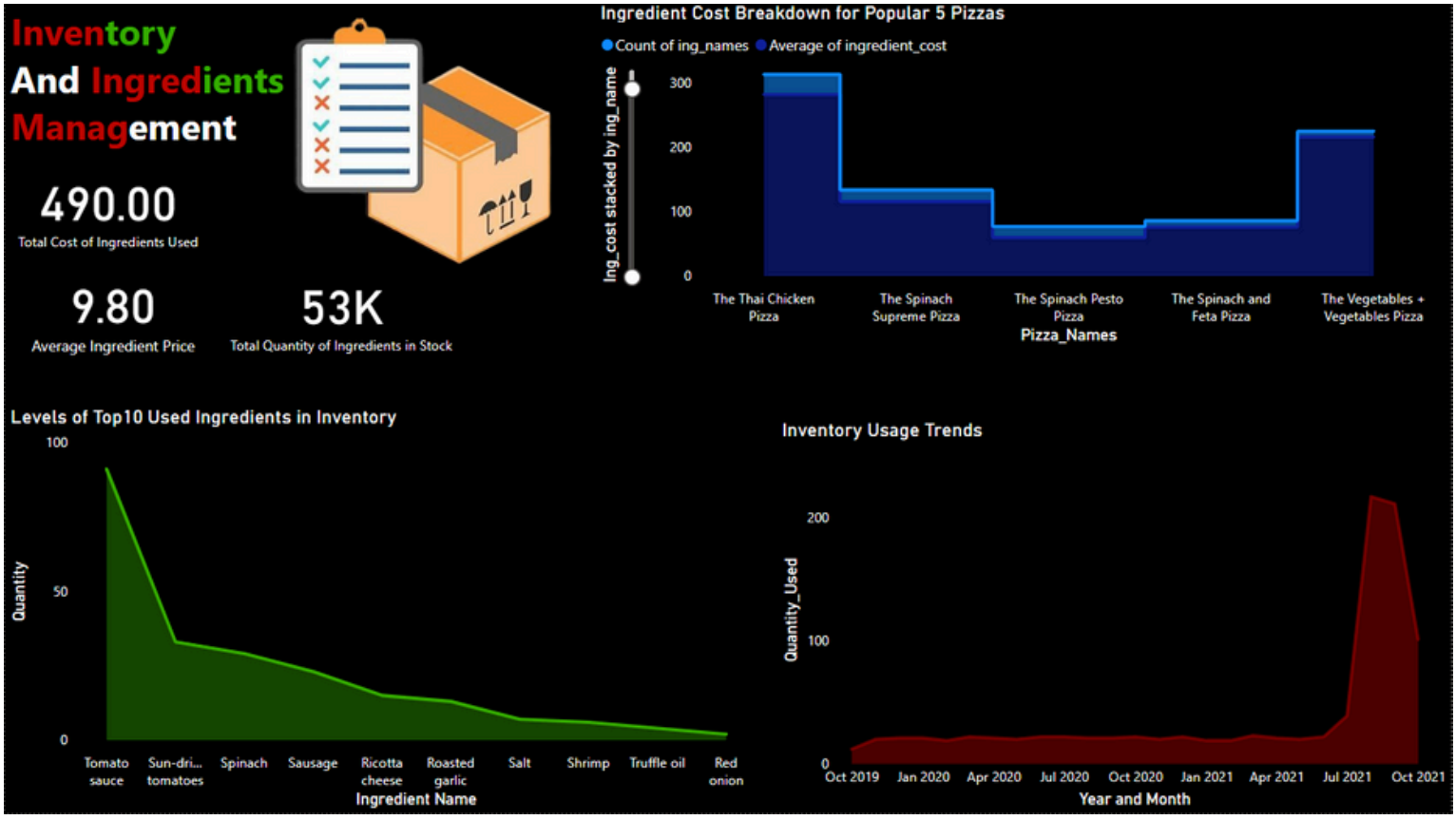
2. Customer Insights:

Insights: The "Customer Insights" section of this dashboard provides a comprehensive analysis of the pizzeria's customer demographics and purchasing behavior. It utilizes various visualizations, including the **Customer Gender Distribution** doughnut chart, the **Age Distribution of Customers** area chart, the **Detailed Customer Information** table, and the **Revenue by Different Pizza Categories** pie chart. These visualizations collectively offer a detailed view of customer characteristics and revenue contributions.



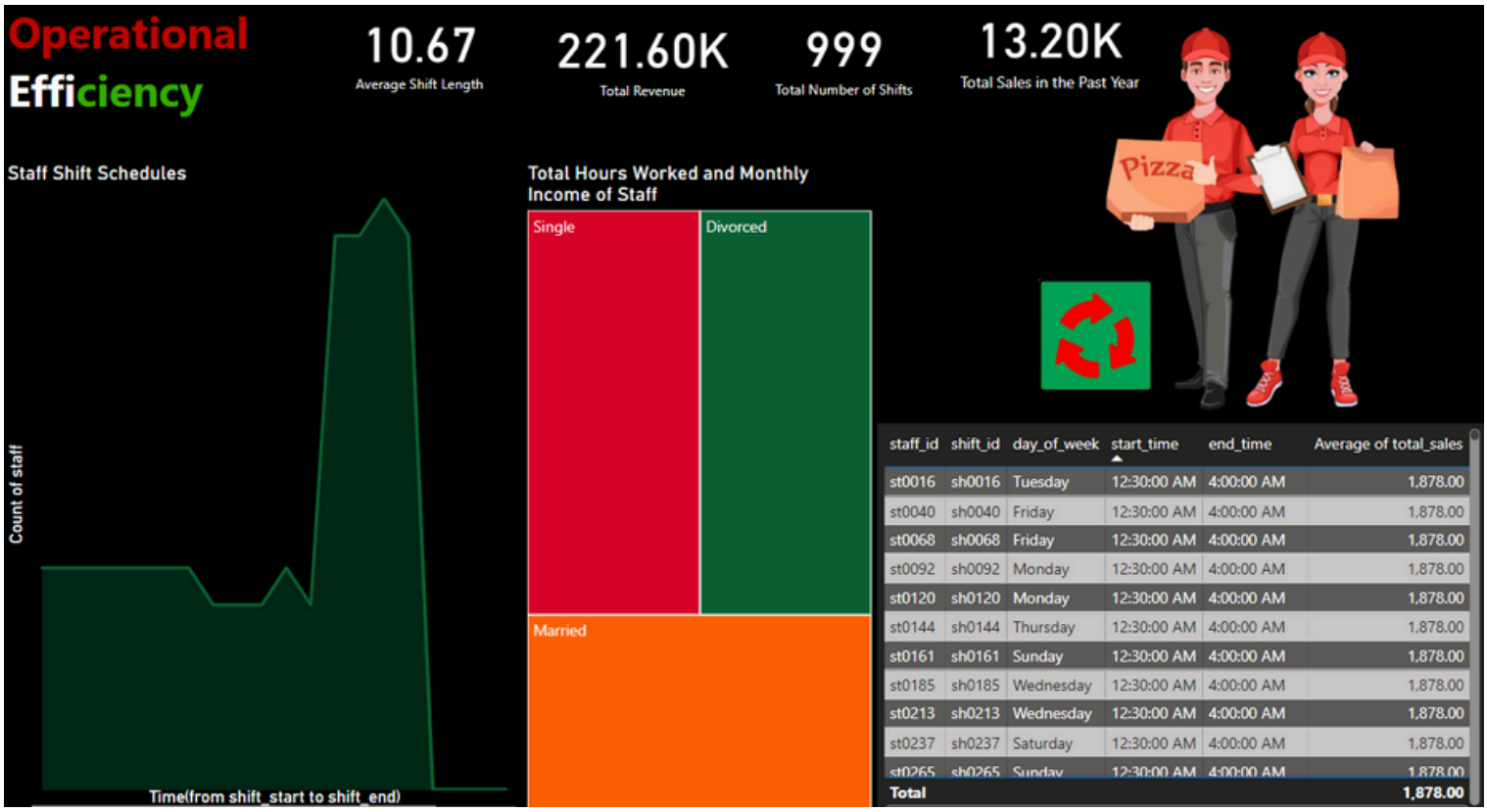
3. Inventory and Ingredients Management:

Insights: The Inventory and Ingredients Management section of this dashboard provides a detailed analysis of the pizzeria's ingredient usage and inventory status. It includes several visualizations, such as the **Total Cost of Ingredients Used**, **Average Ingredient Price**, **Total Quantity of Ingredients in Stock**, the **Ingredient Cost Breakdown for Popular 5 Pizzas** bar chart, the **Levels of Top 10 Used Ingredients in Inventory** area chart, and the **Inventory Usage Trends** area chart. These visualizations offer valuable insights into cost management, inventory levels, and usage trends, aiding in efficient inventory management and cost control.



4. Operational Efficiency:

Insights: The Operational Efficiency Analysis page offers a comprehensive view of key metrics and performance indicators that reflect the efficiency of operations within the organization. This page is meticulously designed to assist management in evaluating and enhancing the effectiveness of staff shifts, analyzing performance variations based on marital status, and identifying overall operational trends.It includes several visualizations, such as **Staff Shift Schedule** Stacked area chart**, Total Hours Worked and Monthly Income of Staff** Treemap, **Detailed Staff and Shift Information** table.



CONCLUSION :

The "Sales Optimization Platform for Pizzeria" uses advanced data analytics and visualizations to provide actionable insights for enhancing sales, customer satisfaction, and operational efficiency. Leveraging MySQL and Power BI, it offers comprehensive analyses in sales, customer behavior, inventory management, and operational efficiency, empowering pizzerias to make data-driven decisions and achieve long-term success.