Pizza Place Sales Proposal

**Consulting Report**

By **MSBA Consulting Student**

Amrit Raj



Table of Contents

EXECUTIVE SUMMARY2

USE CASES3

**User Type 1: Business Managers (Strategic Users) 3**

**User Type 1: Business Managers (Strategic Users) 3**

**BUSINESS RULES4**

**ENTITY RELATIONSHIPS6**

**Entity Relationship (ER) diagram 6**

**Relational Data Schema 6**

**SQL DATABASE7**

**Database overview7**

**Customer Order Form8**

**INSTRUCTION TO ACCESS THE FILEs10**

**SQL database login information 10**

**FORM LOGIN AND ACCESS INFORMATION10**

**TABLEAU PROJECT LOGIN AND ACCESS INFORMATION12**

**PROOF OF CONCEPT14**

**POC OVERVIEW14**

**POC ARCHITECTURE14**

**TESTING APPROACH14**

**KEY FINDINGS15**

**CONCLUSION15**

# **EXECUTIVE SUMMARY**



# The pizza restaurant faced a key business problem: although it handled a large volume of orders throughout the year, management lacked visibility into sales patterns, customer demand behavior, and product-level performance. With over **21,350 orders**, **49,574 pizzas sold**, and total revenue of **$817,860**, decisions such as staffing, pricing, inventory planning, and marketing were made without data-driven insight. The business needed a reliable analytical solution to understand when customers buy the most, which pizza categories drive the highest revenue, and how product preferences vary by size, time, and category.

# To address this, we developed an interactive Pizza Sales Analytics Dashboard using SQL Server and Tableau. The solution consolidated data across four operational tables—orders, order\_details, pizza, and pizza\_types—and transformed them into meaningful visual insights. Monthly sales trends revealed clear seasonality, with revenue peaking at **$72,558 in July** and dipping to **$64,028 in October**. Hourly demand analysis uncovered lunch-time spikes between **12 PM and 1 PM** and again in the early evening (around **17:00–19:00 hrs**), helping identify the periods where customer traffic is highest. Category-level analysis showed that **Classic pizzas** alone generated over **$220,000** comprising **26.9%** of the total revenue generated, making them the restaurant’s most profitable segment. The top-five best-selling pizzas each contributed **$30,000–$45,000** in revenue, highlighting strong menu performers.

# The benefits of this solution are significant. Management can now forecast demand more accurately, optimize staffing during peak hours, and adjust inventory based on real consumption patterns. Marketing efforts can focus on high-performing categories, while underperforming items can be reconsidered or repositioned. The dashboard also supports strategic decisions such as dynamic pricing, targeted promotions, and menu engineering. Ultimately, the organization gains improved operational efficiency, increased revenue opportunities, and better customer satisfaction through data-driven decision making.

By replacing manual, ad‑hoc reporting with an automated, visual KPI framework, the dashboard is expected to improve operational efficiency by an estimated 20–30% through better scheduling and resource allocation, and to support 15–25% revenue growth via targeted promotions, upselling of high‑margin items and sizes, and data‑driven menu optimization.

# Through this solution, the business now has a clear, actionable understanding of its sales performance—something that was previously not possible using raw operational data alone.

# **USE CASES**

This section outlines practical user scenarios demonstrating how two different user groups—Business Managers and Operational Staff—benefit from the Pizza Sales Analytics Dashboard. Each use case highlights how specific features support day-to-day decision-making and operational improvement.

**User Type 1: Business Managers (Strategic Users)**

**Use Case 1: Identifying Monthly Revenue Trends**  
Business managers review the “Total Sales by Month” visualization to understand seasonal demand patterns. By identifying peak months like July and slower periods like September, they make informed decisions about promotional campaigns, pricing strategies, and budgeting for upcoming quarters.

**Use Case 2: Evaluating Product Performance**  
Managers use the “Top 5 Pizzas” and “Revenue by Category” charts to determine which pizzas generate the most revenue. This helps them decide which menu items to highlight, discontinue, or replace. High-performing categories like Classic pizzas guide long-term product strategy.

**Use Case 3: Forecasting Demand for Workforce Planning**  
Using the “Hourly Demand by Category” and “Hourly Demand by Total Orders” charts, managers identify peak operational hours (12 PM–2 PM). This enables them to adjust staffing levels, ensuring the right number of employees are scheduled during high-traffic periods.

**Use Case 4: Strategic Decision-Making Through KPIs**  
Managers analyze KPIs such as Total Revenue, Total Orders, Average Order Value, and Average Quantity per Order to assess business performance. This helps them evaluate growth potential, set targets, and create performance dashboards for leadership reporting.

**User Type 2: Operational Staff (Day-to-Day Users)**

**Use Case 1: Preparing for Daily Inventory Needs**  
Operational staff check category-size matrices and demand breakdowns to anticipate which ingredients will be needed most. For example, if Veggie and Classic pizzas show higher afternoon demand, staff can pre-prep necessary toppings to avoid delays.

**Use Case 2: Real-Time Monitoring of Order Volume**  
Staff view hourly demand insights to plan oven usage, dough preparation, and kitchen workflows. This ensures smooth operations during peak hours and prevents bottlenecks that impact delivery time.

**Use Case 3: Improving Menu Recommendations**  
Frontline employees use top-selling pizza insights to recommend popular items to customers. Knowing which products sell best improves upselling during peak times, enhancing customer satisfaction and driving higher ticket values.

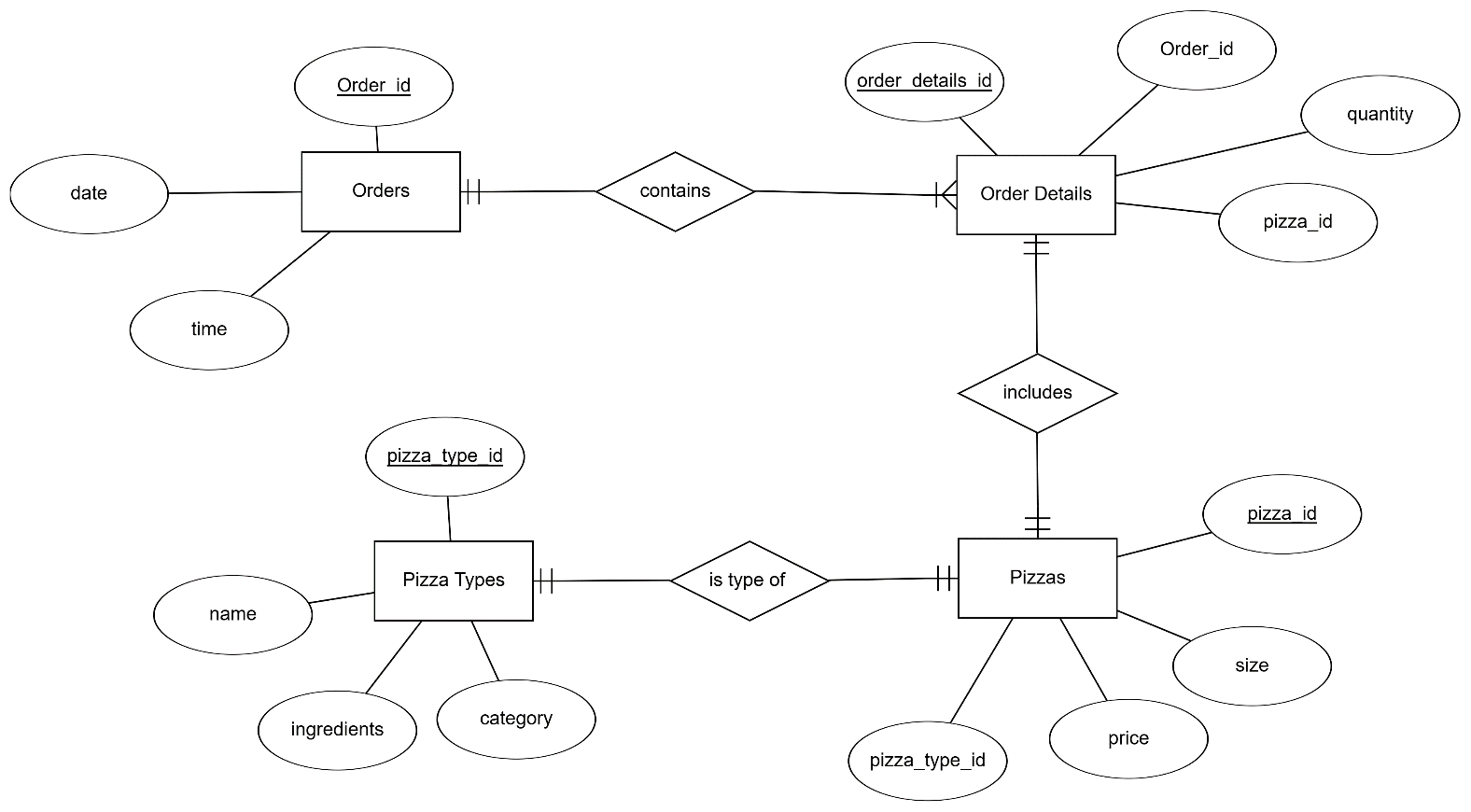
**Use Case 4: Reducing Waste and Managing Stock**  
By checking daily and hourly demand patterns, staff can better estimate how much dough, cheese, vegetables, and meat to prepare. This reduces food waste, prevents stockouts, and keeps operations efficient.

# **BUSINESS RULES**

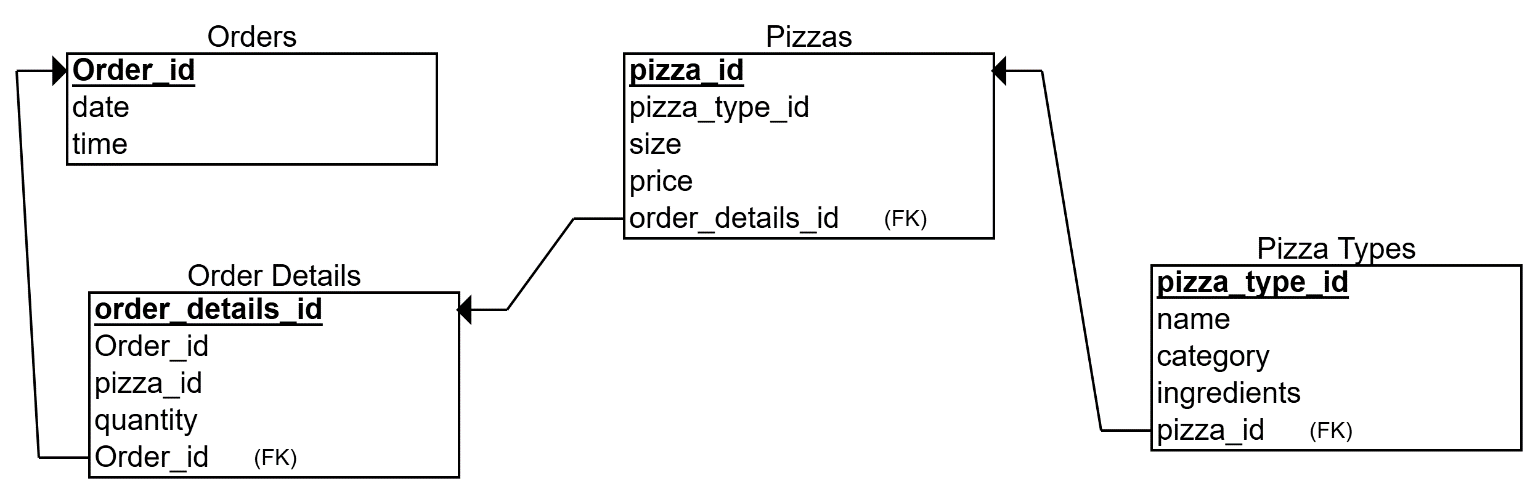
1. Orders and Order Details structure
   * Each order is uniquely identified by Order ID and can contain one or more-line items stored in Order Details table
   * Each order detail row is uniquely identified by Order\_Details\_ID and must reference exactly one OrderID and one PizzaID (no orphan line items).​
2. Product and category modeling
   * Each physical pizza sold is represented by a Pizza entity (e.g., “Thai Chicken, Large”) that belongs to exactly one PizzaType (recipe) and one Category (Classic, Supreme, Chicken, Veggie). Quantity is tracked at the pizza variant level.​
   * Each pizza has exactly one size (S, M, L, XL, XXL), and size plus type uniquely determine a PizzaID at any point in time.​
3. Pricing and revenue rules
   * Unit price is determined by the Pizza (type + size) at the time of sale; OrderDetails stores Unit Price and Quantity, and LineAmount is calculated as Quantity × UnitPrice and stored for reporting consistency.​
   * Order‑level TotalAmount equals the sum of all LineAmount values for that order, optionally adjusted by discounts or coupons captured at the order header level.​
4. Time dimension assumptions
   * Each order is associated with exactly one OrderDate and OrderTime, which are linked to a Date dimension (Year, Quarter, Month, Day) and optionally a Time dimension (hour of day) for trend and peak‑hour analysis.​
   * All sales in the analytical model use the store’s local time zone; late‑night orders after midnight belong to their actual calendar date, not the prior business day.​
5. Store and channel modeling
   * Each order is placed at exactly one Store (location) and through one Channel (dine‑in, delivery, takeout, online), modeled as separate dimensions for performance comparison.​
   * A store must exist and be active on the order date for the order to be considered valid in the fact table.​
6. Customer and loyalty assumptions
   * When customer data is available, each order can optionally reference one CustomerID; anonymous walk‑in orders are stored with a special “Unknown” customer in the customer dimension.​
   * Customer attributes such as loyalty tier or segment are treated as slowly changing dimensions; historical orders retain the attributes valid at the time of the order.​
7. Data quality and uniqueness rules
   * OrderID, PizzaID, StoreID, and other business keys must be unique and non‑null in their respective dimension tables; referential integrity is enforced between fact and dimensions.​
   * Negative quantities are not allowed; returns or cancellations are modeled as separate transactions (e.g., a reversal entry with its own OrderID) rather than negative line items.​

# **ENTITY RELATIONSHIPS**

Entity Relationship (ER) diagram



Relational Data Schema



# **SQL DATABASE**

**Database overview**

A relational SQL database was created using a physical data model to store pizza sales and customer order information. The schema includes five core tables: orders, order\_details, pizzas, pizza\_types, and a learning table customer\_orders used to capture new orders from a Microsoft Form.​

orders

The orders table stores one record per order and acts as the header table. Typical columns include a surrogate primary key order\_id, the order date and time, and other order‑level attributes such as customer or store information. This table is referenced by order\_details through a foreign key on order\_id.​

order\_details

order\_details is the line‑item table that records each pizza included in an order. It contains an order\_details\_id primary key, a foreign key order\_id pointing to orders, pizza\_id linking to the pizzas table, and a quantity field showing how many of that pizza were ordered. This structure allows one order to contain multiple pizzas and supports granular sales analysis by item.​

pizzas and pizza\_types

The pizzas table defines individual pizza SKUs with attributes such as pizza\_id, size, price, and a foreign key that links to pizza\_types. pizza\_types stores higher‑level information about each pizza type, including its name, category (for example, vegetarian or meat), and ingredients. Together, these two tables normalize product data and enable queries that join sales with pizza category and recipe information.​

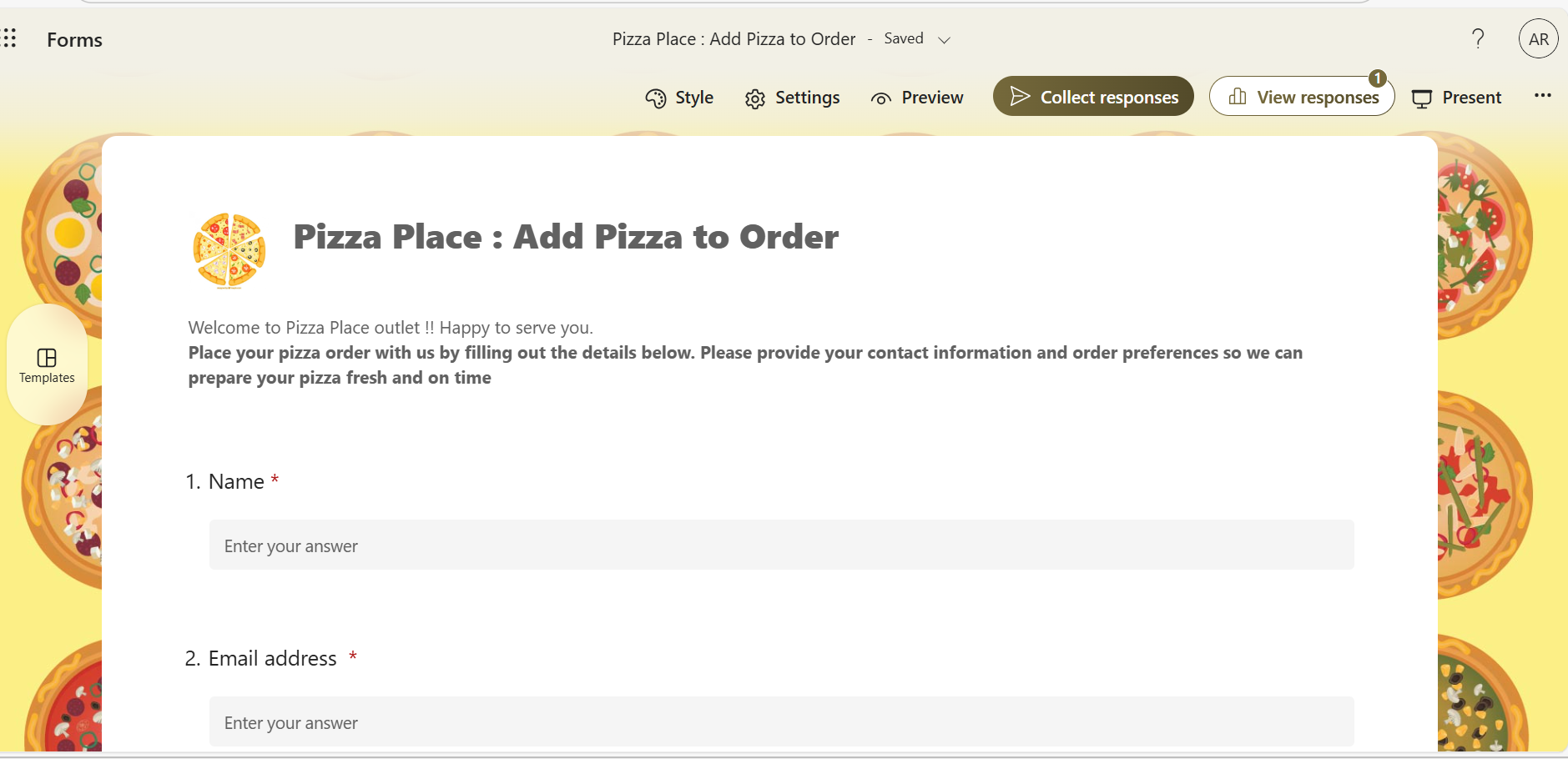
customer\_orders (learning table)

A separate customer\_orders table was added for learning and integration with Microsoft Forms. Each row captures a single pizza order placed through the form, including customer\_order\_id (identity key), customer name, mail\_id, order\_number, pizza\_id, quantity, and order\_time. This table does not modify the original schema but provides a clean target for automated inserts from Power Automate while preserving all necessary order details.

**customer\_orders table structure**

| **Column name** | **Data type** | **Description** |
| --- | --- | --- |
| customer\_order\_id | INT IDENTITY | Surrogate primary key for each form submission |
| name | VARCHAR(100) | Customer name entered in the form |
| mail\_id | VARCHAR(255) | Customer email address |
| order\_number | INT | Order number captured from the form |
| pizza\_id | VARCHAR(50) | Code of the pizza selected (matches menu ID) |
| quantity | INT | Number of pizzas ordered |
| order\_time | DATETIME | Date and time when the order is recorded |

**Customer Order Form:  
Link:** [Pizza Place : Add Pizza to Order – Fill out form](https://forms.microsoft.com/Pages/ResponsePage.aspx?id=W9229i_wGkSZoBYqxQYL0r7haT6USN9PtLfZdI4CFWpUQUI5M1dOWVk1NFZMSzVLN0swQ1JWQzY1VC4u)

****

The form was integrated with the **Azure SQL Server database** using **Power Automate**, enabling real-time data transfer. Whenever a customer submits the form, the responses are automatically inserted into the customer order table within the database. This automation eliminates manual data entry and ensures that customer records are instantly available for menu preparation, packaging, and delivery time.

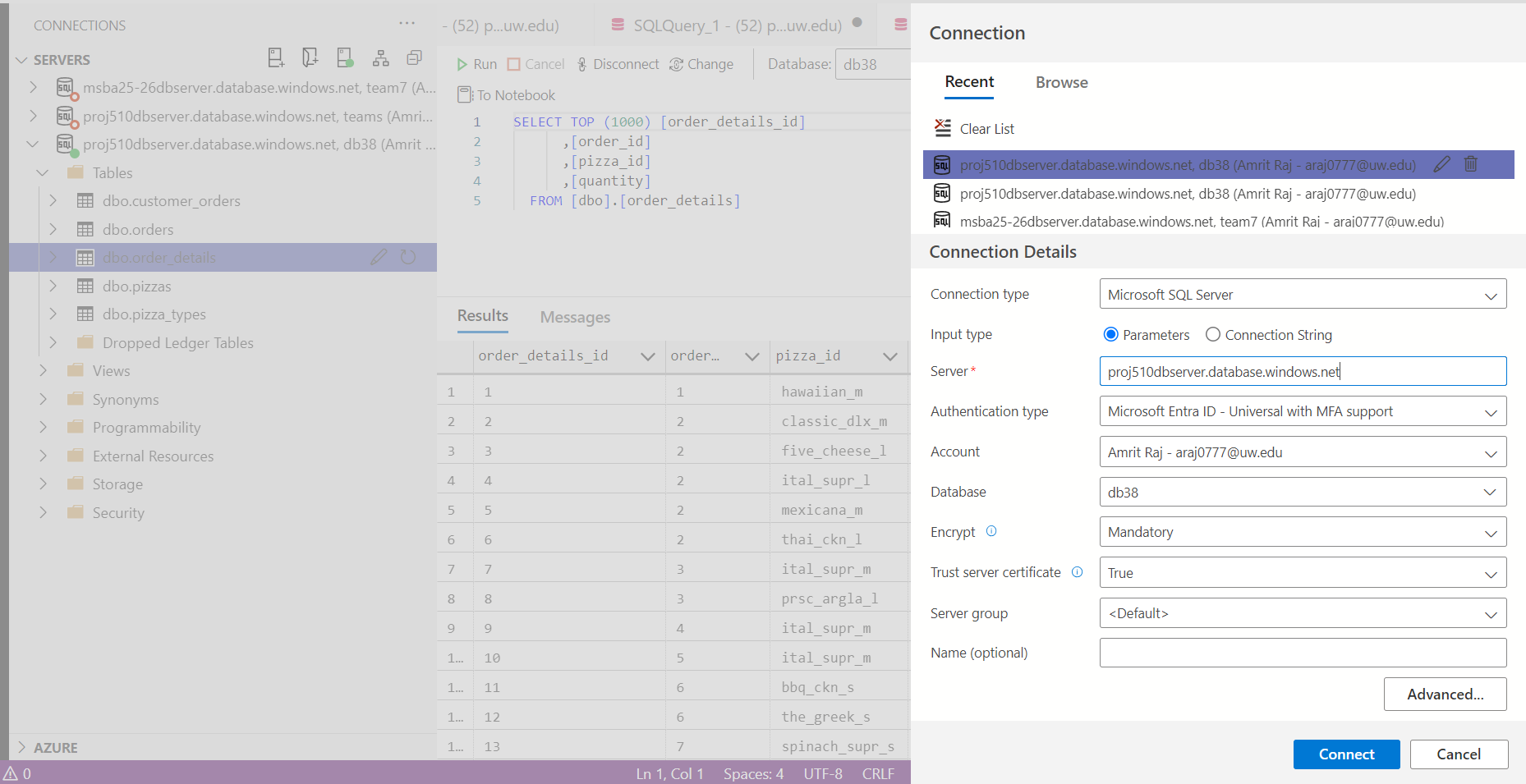
A Microsoft Form named Pizza Place Order was created as a front-end for the customer\_orders SQL table. The form collects customer and order details through the following fields:

* Name (Text) – captures the customer’s name.
* Email address (Text) – captures the customer’s mail ID, validated as an email format.
* Order number (Text with “Number / Whole number” restriction) – stores the numeric order\_number value used in the database.
* Pizza ID / Pizza selection (Choice question listing available pizza IDs) – maps directly to the pizza\_id column.
* Quantity (Text with “Number / Whole number” restriction) – stores how many pizzas are ordered.
* Order time (Date/Time question, or omitted so the database fills order\_time with the current timestamp automatically).

**INSTRUCTION TO ACCESS THE FILES**

1. **SQL database login information**

The SQL Server database is hosted at **server name : proj510dbserver.database.windows.net** with database [db38]. Users connect using SQL authentication with the username (araj0777@uw.edu) and password provided separately. The connection string is configured in SQL Server Management Studio (or the application) and the “Test Connection” button returns a successful result, confirming that the login instructions are working.



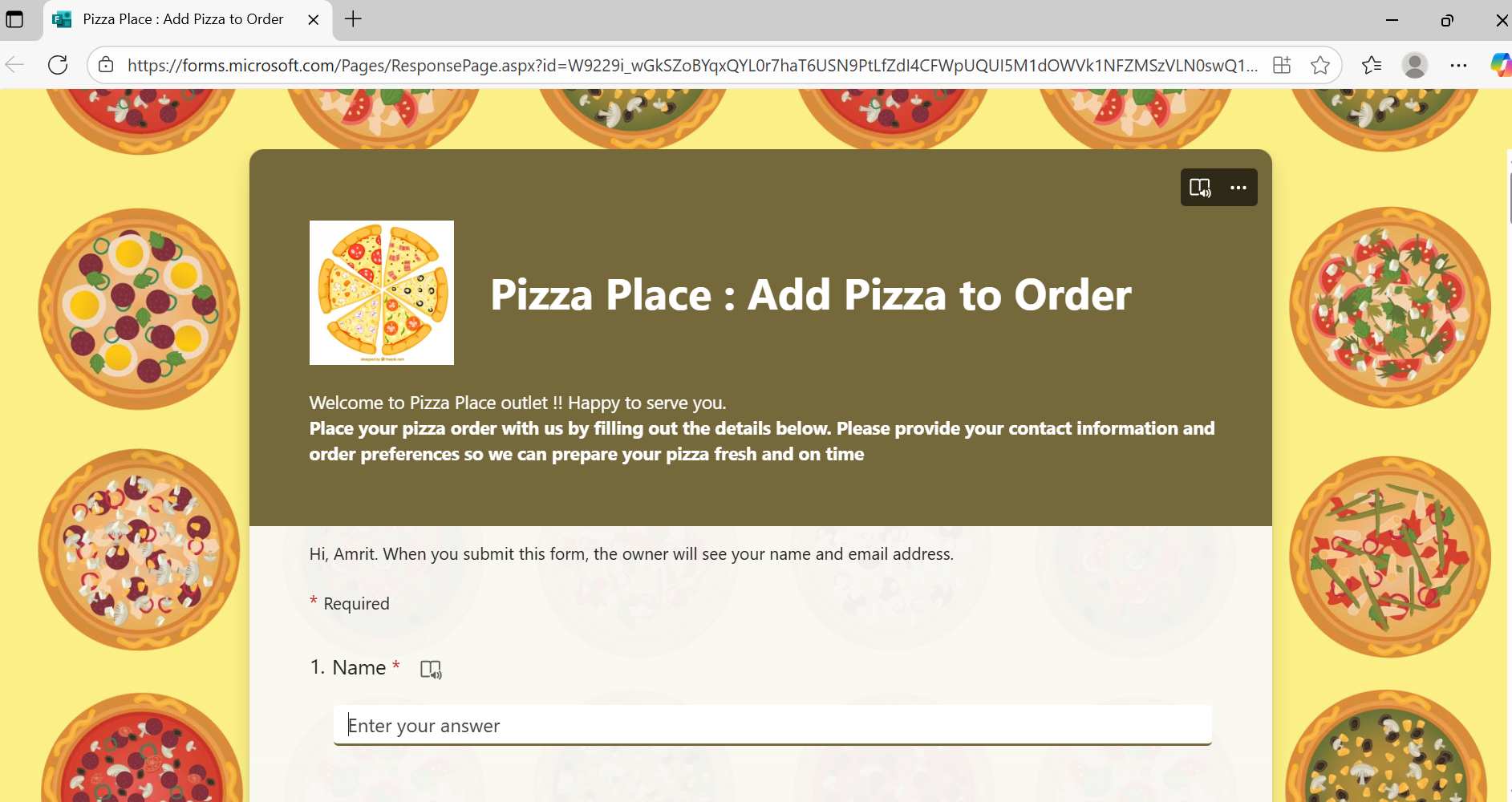
Data set: <https://www.kaggle.com/datasets/aaditya555/pizza-sales-dataset>

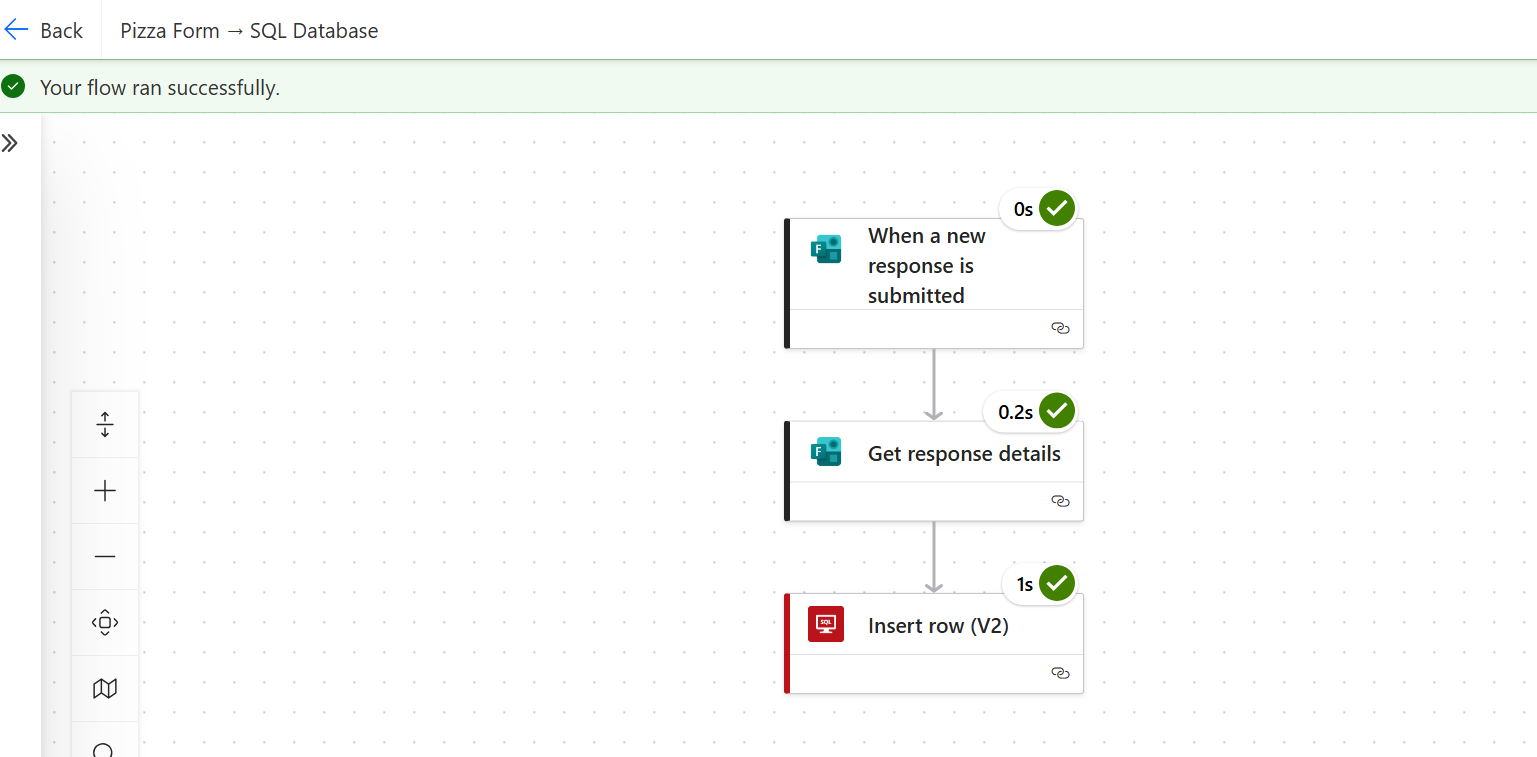
**Server name**: proj510dbserver.database.windows.net  
Database: db38

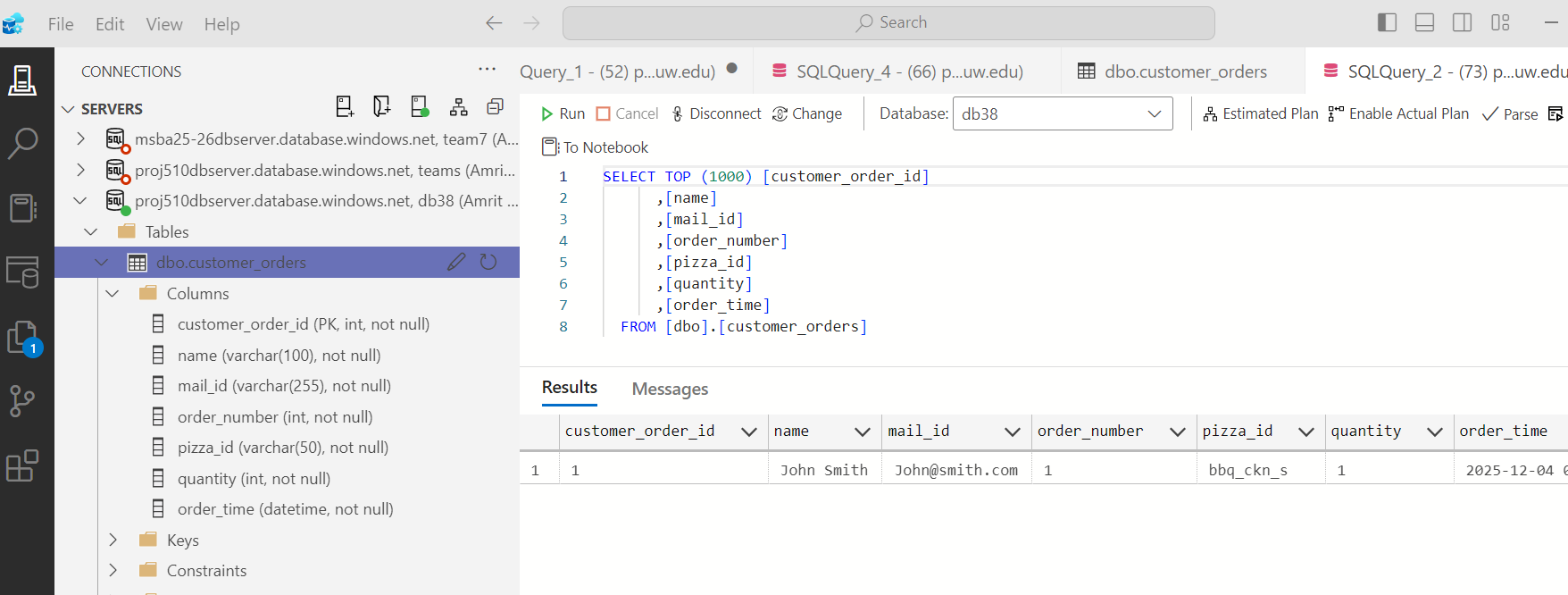
**UW NetID**: araj0777

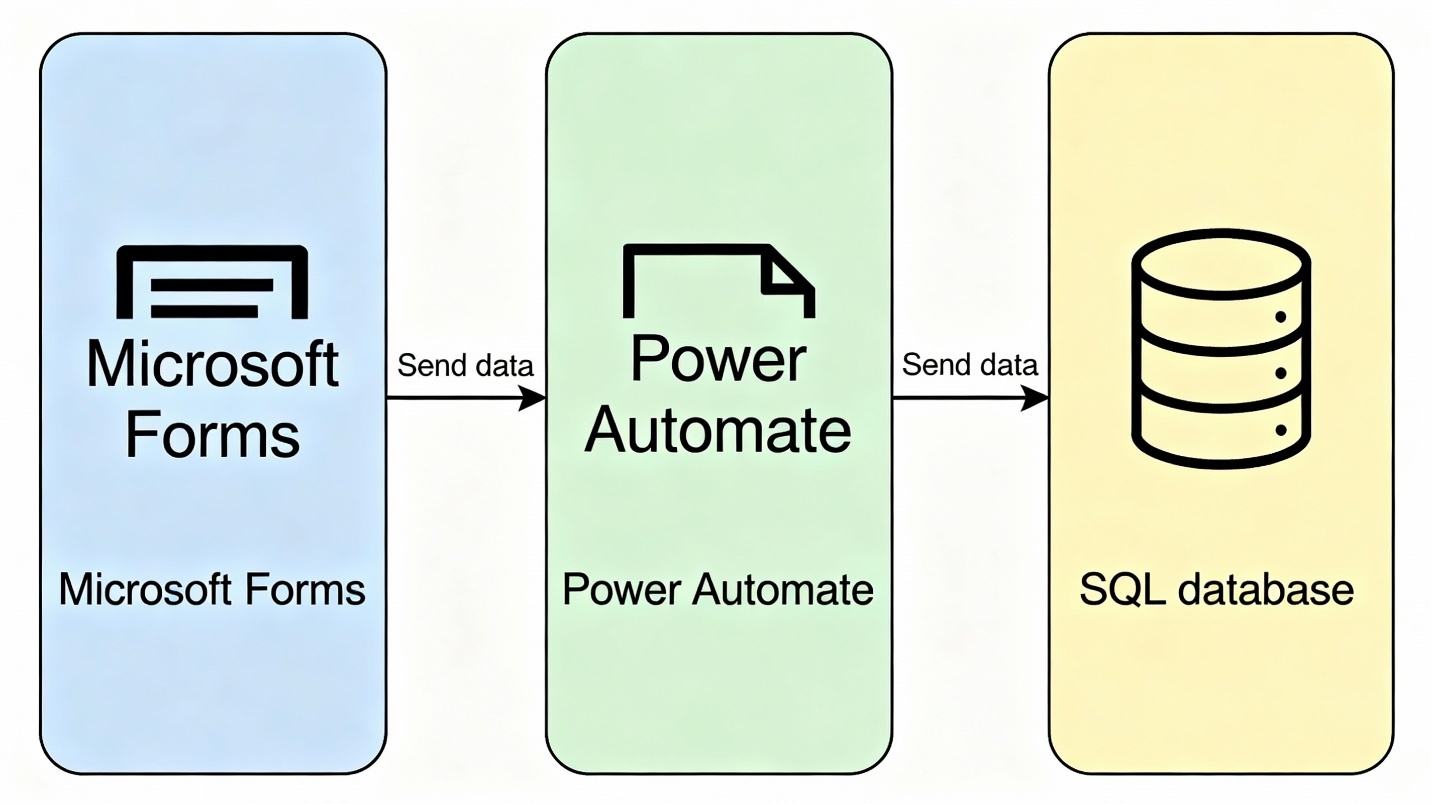
1. **Form login and access information**

The Pizza Place Order form is published in Microsoft Forms under the account [account/email]. Authorized users sign in to Microsoft 365 with this account (or their assigned accounts) and open the form via the shared link [form URL : [Pizza Place : Add Pizza to Order – Fill out form](https://forms.microsoft.com/r/tWs2vuY8vN)]. Submitting the form creates a new entry that is passed to the SQL database through the configured Power Automate flow, confirming that the form access instructions are working.



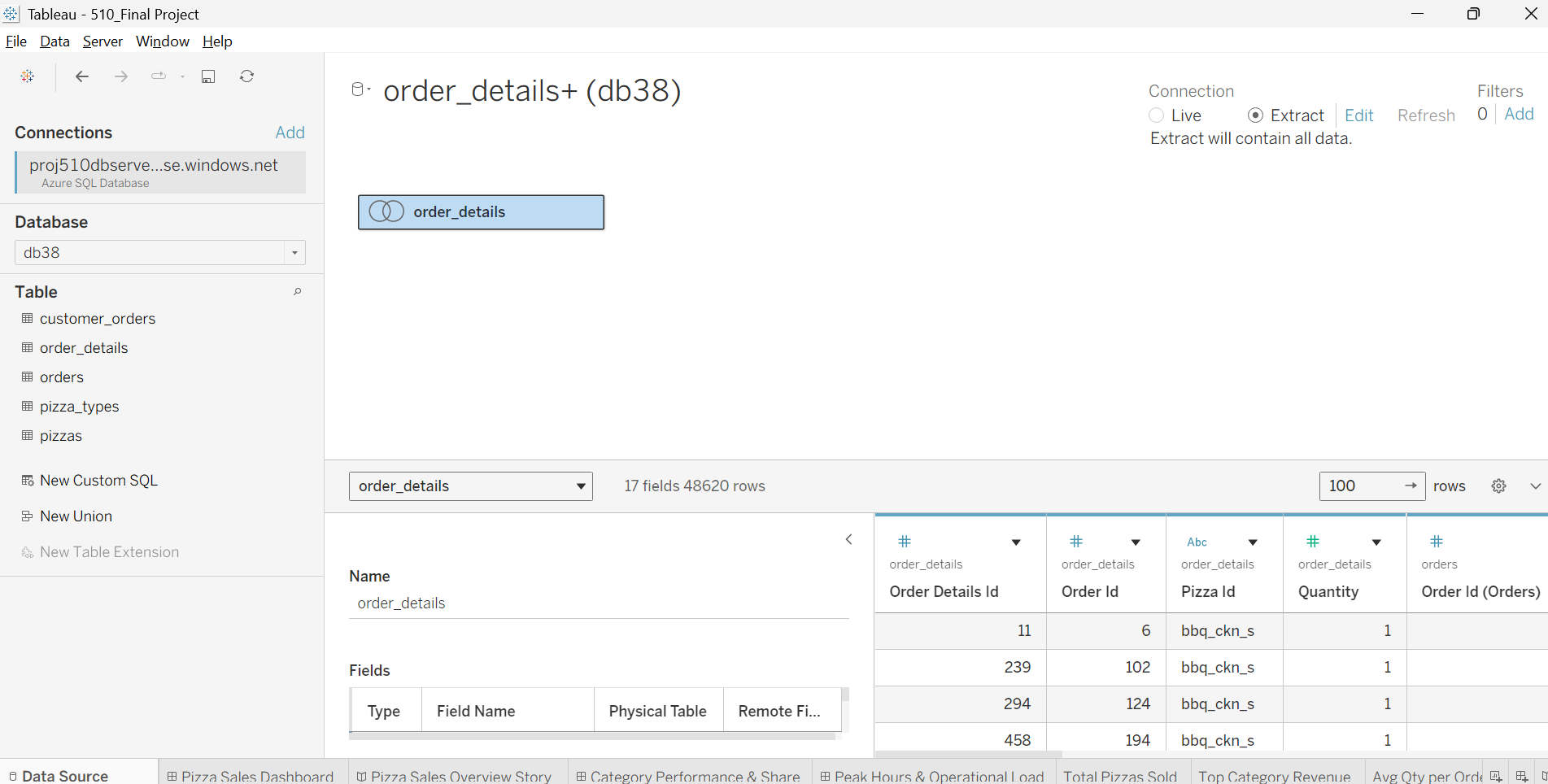
****

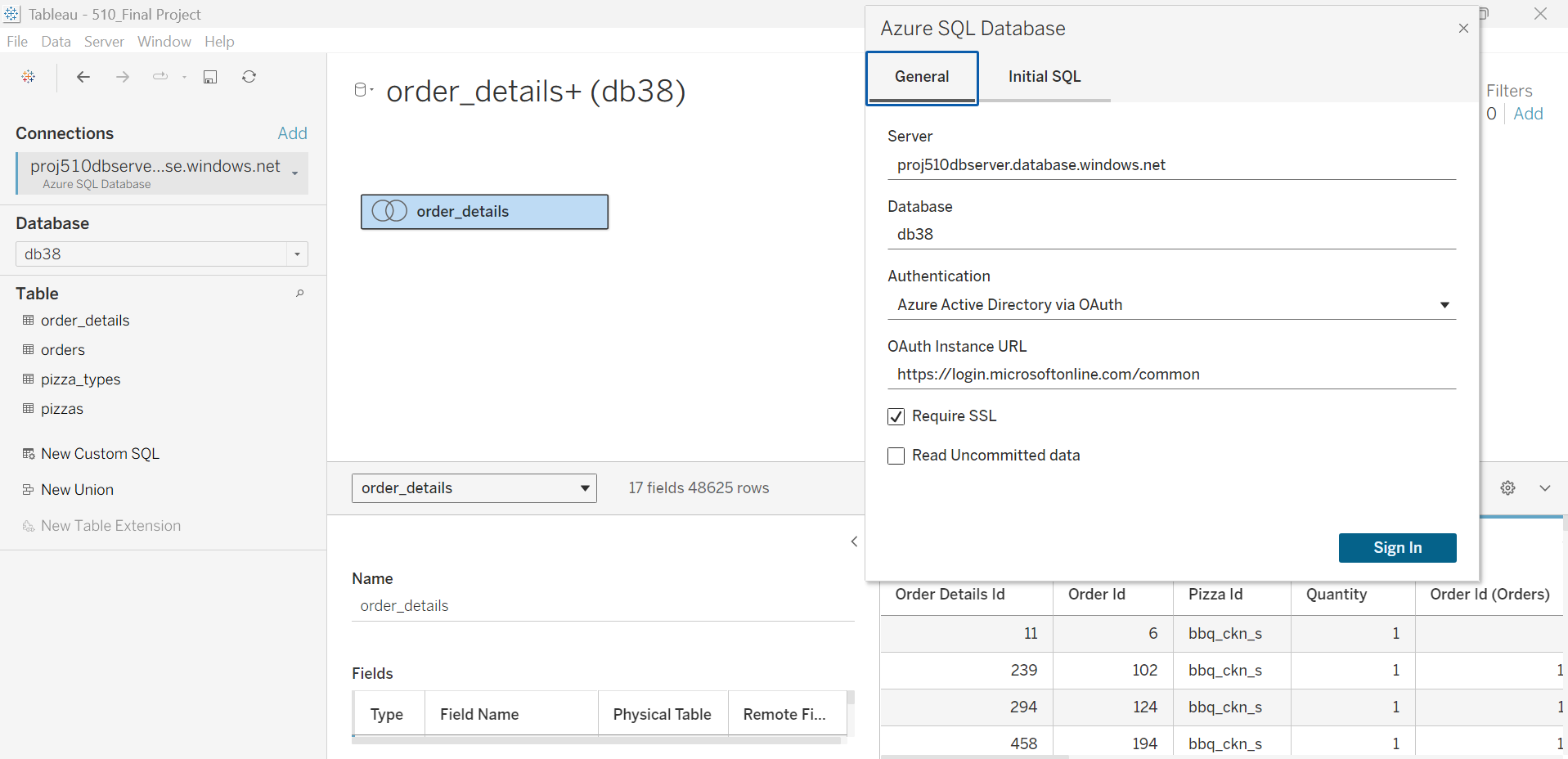




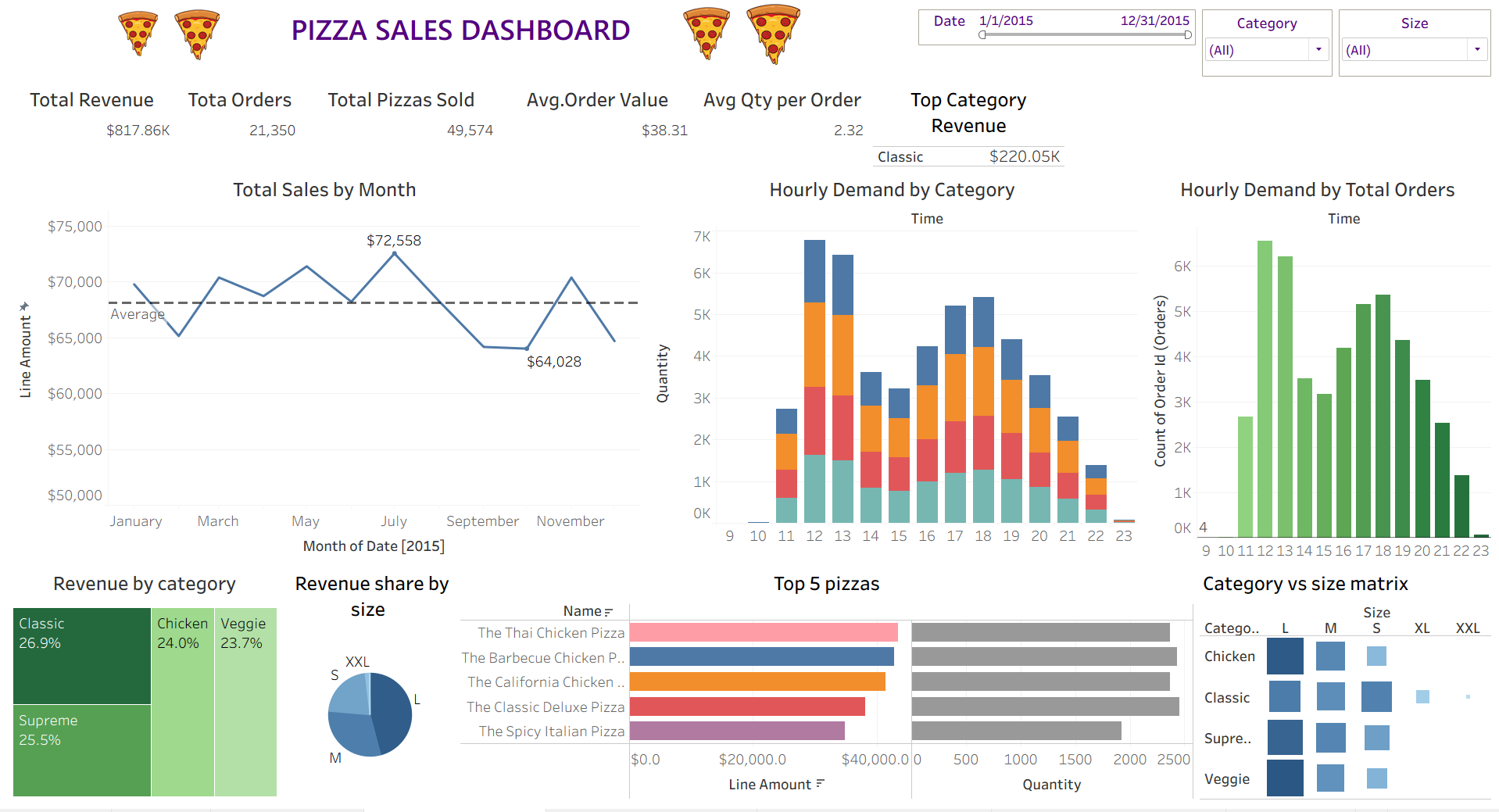
1. **Tableau project login and access information**

The Tableau workbook is stored in [Tableau Cloud [**https://public.tableau.com/app/profile/amrit.raj3866/viz/FinalProject\_510\_17649706830990/PizzaSalesDashboard?publish=yes**](https://public.tableau.com/app/profile/amrit.raj3866/viz/FinalProject_510_17649706830990/PizzaSalesDashboard?publish=yes)] and storyboard ([**https://public.tableau.com/app/profile/amrit.raj3866/viz/FinalProject\_510\_17649706830990/PizzaSalesOverviewStory?publish=yes**](https://public.tableau.com/app/profile/amrit.raj3866/viz/FinalProject_510_17649706830990/PizzaSalesOverviewStory?publish=yes)). Users open Tableau Desktop (or Tableau Server in a browser), sign in with their Tableau credentials, and select the Pizza Sales dashboard. The workbook connects to the same SQL database using the configured data source and successfully refreshes, confirming that the Tableau access instructions are working***.***

****

****

**KPI Dashboard , Storyboard and Data Visualization**

****



**PROOF OF CONCEPT**

**Information for client to use and test the POC**

**Proof of Concept (POC) – Overview**

This Proof of Concept demonstrates an automated end-to-end data pipeline for Pizza Place orders, integrating Microsoft Forms, Power Automate, SQL Server, and Tableau. The solution enables real-time order capture, automated data loading, and dynamic visualization for business insights.

**POC Architecture**

The workflow consists of four connected components:

1. Microsoft Forms – Used by customers to submit new pizza orders.
2. Power Automate Flow – Automatically triggers when a form is submitted, processes the response, and inserts the order into SQL Server.
3. SQL Database (dbo.customer\_orders) – Stores the order data for analytics and reporting.
4. Tableau Dashboard – Visualizes updated orders and KPIs for decision-making.

This architecture ensures seamless data collection, processing, storage, and reporting.

**Testing Approach**

A high-level testing framework was followed to validate the POC:

* **Functional Testing:**  
  Verified that every form submission triggers the flow and inserts accurate data into SQL Server.
* **Integration Testing:**  
  Confirmed smooth interaction between Forms → Power Automate → SQL → Tableau.
* **Data Validation:**  
  Checked record completeness, field mapping, and value accuracy in the SQL table.
* **Dashboard Validation:**  
  Ensured that refreshed data sources in Tableau accurately reflected newly added orders.

**Key Findings**

* Power Automate reliably captured and inserted orders with no failed runs during testing.
* Data fields (e.g., customer name, size, quantity, amount) matched expected values.
* Tableau visuals updated correctly after refresh, confirming proper backend connectivity.
* The POC successfully demonstrates real-time integration of Microsoft business tools.

**Conclusion**

The POC meets all intended objectives. It proves that the automated pipeline is functional, scalable, and ready for extended testing or production enhancement. The system allows the client to capture orders instantly and visualize business metrics efficiently. This method can be adopted for expansion of new outlets /branch within the city.