BUSA8090 - Data and Visualisation for Business
Assignment 1

Urban Eats Café Chain: Analytics and Database Design Report

Student Name: Muntasir Md Nafis

Student ID: 48312932

Part A – Database Design and Development

1.1 Entity Identification

Entity Name	Entity Description
Outlet	List of store locations with their name, address, phone, and email.
Customer	Registered customers with names and contact details (plus optional address) used to tie reservations and orders.
Product_Category	List of product categories with their names and short descriptions.
Product	Individual menu items with price, linked to a product category for analysis and availability.
Shift	Scheduled work periods per outlet with start/end times and maximum staffing capacity.
Reservation	Customer bookings at outlets capturing date/time, number of people and reservation status.
Orders	Customer purchases at outlets with order date, type (dine-in/takeaway/delivery) and current status.
Payment	Financial records for each order including amount, method and payment status.
Staff	Employee directory with role/designation, employment status, manager hierarchy and outlet assignment.
Outlet_Product	Mapping of which products are available or unavailable at each outlet.
Order_Product	Links each order to its products with quantity and price.
Staff_Shift	Assignment of staff to specific shifts with role performed and hours worked.

1.2 Identify Attributes, Primary Keys and Foreign Keys

Entity Name	Attributes
OUTLET	Outlet_ID **(PK)**, Outlet_Name, Address, Phone, Email
CUSTOMER	Customer_ID **(PK)**, First_Name, Last_Name, Email, Contact, Customer_Address
PRODUCT_CATEGORY	Product_Category_ID **(PK)**, Product_Category_Name, Product_Category_Description

PRODUCT	Product_ID **(PK)**, Product_Name, Price, Category_ID **(FK → Product_Category.Product_Category_ID)**
SHIFT	Shift_ID **(PK)**, Outlet_ID **(FK → Outlet.Outlet_ID)**, Start_Time, End_Time, Max_Staff
RESERVATION	Reservation_ID **(PK)**, Outlet_ID **(FK → Outlet.Outlet_ID)**, Customer_ID **(FK → Customer.Customer_ID)**, Date, Time, Number_of_People, Status
ORDERS	Order_ID **(PK)**, Outlet_ID **(FK → Outlet.Outlet_ID)**, Customer_ID **(FK → Customer.Customer_ID)**, Order_Date, Order_Type, Order_Status
PAYMENT	Payment_ID **(PK)**, Order_ID **(FK → Orders.Order_ID)**, Payment_Amount, Payment_Method, Payment_Status
STAFF	Staff_ID **(PK)**, First_Name, Last_Name, Designation, Employment_Status, Manager_ID **(FK → Staff.Staff_ID)**, Outlet_ID **(FK → Outlet.Outlet_ID)**
OUTLET-PRODUCT	Product_ID **(PK, FK → Product.Product_ID)**, Outlet_ID **(PK, FK → Outlet.Outlet_ID)**, Outlet_Status
ORDER-PRODUCT	Order_ID **(PK, FK → Orders.Order_ID)**, Product_ID **(PK, FK → Product.Product_ID)**, Quantity, Price
STAFF-SHIFT	Staff_ID **(PK, FK → Staff.Staff_ID)**, Shift_ID **(PK, FK → Shift.Shift_ID)**, Staff_Role, Working_Hours

1.3 Relationship Definition and Cardinality Analysis

The relationship between Outlet and Order is one-to-many, as each outlet can have many orders while every order belongs to exactly one outlet. Similarly, the relationship between Customer and Order is also one-to-many, because a customer can place multiple orders, but each order is tied to a single customer. Between Order and Payment, the relationship is one-to-many since an order may generate multiple payment records (including partial payments, refunds, or failures), but each payment refers to only one order.

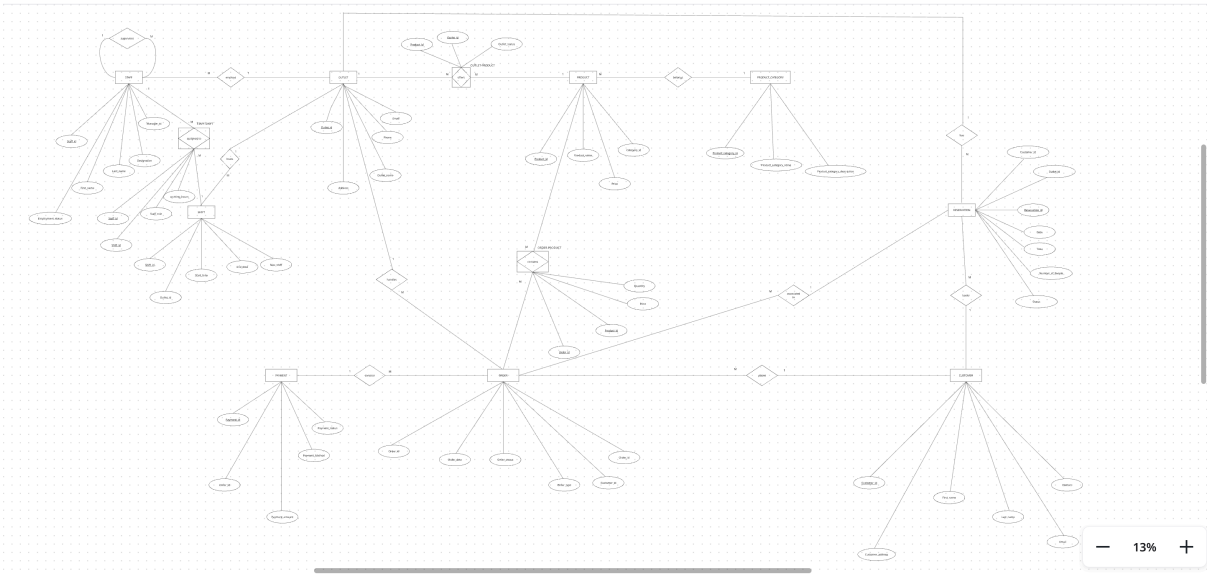
For Reservations, the link to both Outlet and Customer is one-to-many. An outlet can host many reservations and a customer can book multiple reservations, but each reservation record belongs to one outlet and one customer only.

The Product and Product_Category entities are in a one-to-many relationship, with each category containing many products, while each product belongs to exactly one category. To capture order details, the Orders and Products relationship is implemented through Order_Product(associative entity), which creates a many-to-many relationship. An order can contain multiple products, and a product can appear in many different orders.

The availability of menu items across locations is handled by Outlet_Product(associative entity), which establishes a many-to-many relationship between outlets and products. Each outlet may list several products, and each product can be offered in multiple outlets.

In staffing, the relationship between Outlet and Shift is one-to-many, as an outlet schedules multiple shifts, but each shift belongs to one outlet. The Staff_Shift(associative entity) table defines a many-to-many relationship between staff and shifts, where staff members can cover multiple shifts, and each shift can include multiple staff members. Additionally, within the Staff table, there is a one-to-many uniary relationship: one staff member can act as a manager for many others, but each staff member has at most one direct manager.

1.4 ERD diagram



1.5 Table implementation

```
# Creating OUTLET Table
CREATE TABLE Outlet (
  Outlet_ID     INT         PRIMARY KEY,
  Outlet_Name   VARCHAR(100)  NOT NULL,
  Address       VARCHAR(200)  NOT NULL,
  Phone         VARCHAR(20),
  Email         VARCHAR(120)
);

# Creating CUSTOMER Table
CREATE TABLE Customer (
  Customer_ID   INT         PRIMARY KEY,
  First_Name    VARCHAR(60)   NOT NULL,
  Last_Name     VARCHAR(60)   NOT NULL,
  Email         VARCHAR(120),
  Contact       VARCHAR(20)
);

ALTER TABLE Customer
ADD Customer_Address VARCHAR(200);


# Creating PRODUCT_CATEGORY Table
CREATE TABLE Product_Category (
  Product_Category_ID     INT         PRIMARY KEY,
  Product_Category_Name   VARCHAR(100)  NOT NULL,
```

```
  Product_Category_Description VARCHAR(255)
);

# Creating PRODUCT Table
CREATE TABLE Product (
  Product_ID    INT PRIMARY KEY,
  Product_Name  VARCHAR(120) NOT NULL,
  Price        DECIMAL(10,2) NOT NULL,
  Category_ID   INT NOT NULL,
  FOREIGN KEY (Category_ID) REFERENCES Product_Category(Product_Category_ID)
);

# Creating SHIFT Table
CREATE TABLE Shift (
 Shift_ID   INT PRIMARY KEY,
 Outlet_ID  INT NOT NULL,
 Start_Time TIME NOT NULL,
 End_Time   TIME NOT NULL,
 Max_Staff  INT NOT NULL,
 FOREIGN KEY (Outlet_ID) REFERENCES Outlet(Outlet_ID)
);

# Creating RESERVATION Table
CREATE TABLE Reservation (
 Reservation_ID   INT PRIMARY KEY,
 Outlet_ID       INT NOT NULL,
 Customer_ID     INT NOT NULL,
 Date            DATE NOT NULL,
```

```
 Time           TIME NOT NULL,
 Number_of_People  INT NOT NULL,
 Status         VARCHAR(20),
 FOREIGN KEY (Outlet_ID)   REFERENCES Outlet(Outlet_ID),
 FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)
);

# Creating ORDER Table
CREATE TABLE Orders (
 Order_ID    INT PRIMARY KEY,
 Outlet_ID   INT NOT NULL,
 Customer_ID  INT NOT NULL,
 Order_Date   DATE NOT NULL,
 Order_Type   VARCHAR(20) NOT NULL,
 Order_Status VARCHAR(20) NOT NULL,
 FOREIGN KEY (Outlet_ID)   REFERENCES Outlet(Outlet_ID),
 FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)
);

# Creating PAYMENT Table
CREATE TABLE Payment (
 Payment_ID     INT PRIMARY KEY,
 Order_ID       INT NOT NULL,
 Payment_Amount  DECIMAL(10,2) NOT NULL,
 Payment_Method  VARCHAR(20),
 Payment_Status  VARCHAR(20),
 FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID)
);
```

```
# Creating STAFF Table
CREATE TABLE Staff (
   Staff_ID         INT PRIMARY KEY,
   First_Name       VARCHAR(60) NOT NULL,
   Last_Name        VARCHAR(60) NOT NULL,
   Designation      VARCHAR(50) NOT NULL,
   Employment_Status  VARCHAR(20) NOT NULL,
   Manager_ID       INT ,
   Outlet_ID        INT NOT NULL,
   FOREIGN KEY (Outlet_ID)  REFERENCES Outlet(Outlet_ID),
   FOREIGN KEY (Manager_ID) REFERENCES Staff(Staff_ID)
);



# Creating OUTLET-PRODUCT Table
CREATE TABLE Outlet_Product (
 Product_ID    INT NOT NULL,
 Outlet_ID     INT NOT NULL,
 Outlet_Status VARCHAR(20),
 PRIMARY KEY (Product_ID, Outlet_ID),
 FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
 FOREIGN KEY (Outlet_ID)  REFERENCES Outlet(Outlet_ID)
);



# Creating ORDER-PRODUCT Table
```

```
CREATE TABLE Order_Product (
 Order_ID   INT NOT NULL,
 Product_ID INT NOT NULL,
 Quantity   INT NOT NULL,
 Price      DECIMAL(10,2) NOT NULL,
 PRIMARY KEY (Order_ID, Product_ID),
 FOREIGN KEY (Order_ID)   REFERENCES Orders(Order_ID),
 FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)
);

# Creating STAFF-SHIFT Table
CREATE TABLE Staff_Shift (
 Staff_ID     INT NOT NULL,
 Shift_ID     INT NOT NULL,
 Staff_Role    VARCHAR(40),
 Working_Hours DECIMAL(5,2),
 PRIMARY KEY (Staff_ID, Shift_ID),
 FOREIGN KEY (Staff_ID) REFERENCES Staff(Staff_ID),
 FOREIGN KEY (Shift_ID) REFERENCES Shift(Shift_ID)
);
```

1.6 Dummy Data insertion

```
# Inserting Dummy Data into OUTLET Table
```

```
INSERT INTO Outlet (Outlet_ID, Outlet_Name, Address, Phone, Email) VALUES

(1, 'Urban Eats Central', '100 George St, Sydney NSW 2000', '02 8000 1001',
'central@urbaneats.com.au'),

(2, 'Urban Eats Harbour', '22 Wharf Rd, Sydney NSW 2000',  '02 8000 1002',
'harbour@urbaneats.com.au'),

(3, 'Urban Eats Campus',  '15 University Ave, Macquarie NSW 2109', '02 8000 1003',
'campus@urbaneats.com.au');


# Inserting Dummy Data into CUSTOMER Table


INSERT INTO Customer (Customer_ID, First_Name, Last_Name, Email, Contact)
VALUES

(101, 'Ava',   'Brown',   'ava.brown@gmail.com',      '0411 234 567'), -- Central repeater +
reservation (Challenges Associated)

(102, 'Liam',   'Ng',     'liam.ng@outlook.com',      '0412 345 678'), -- one-timer (at risk)

(103, 'Noah',   'Khan',   'noah.khan@gmail.com',      '0413 456 789'), -- Harbour delivery
loyalist

(104, 'Mia',   'Chen',   'mia.chen@proton.me',       '0414 567 890'), -- Harbour dine-in +
reservation

(105, 'Olivia',  'Taylor',  'olivia.taylor@gmail.com',   '0415 678 901'), -- Campus no-show
risk

(106, 'Lucas',  'White',   'lucas.white@outlook.com',   '0416 789 012'), -- Central
takeaway repeater

(107, 'Grace',  'Park',   'grace.park@gmail.com',     '0417 890 123'), -- Central occasional
(desserts)

(108, 'Ethan',  'Rao',    'ethan.rao@gmail.com',      '0418 901 234'), -- Harbour cold-
drinks fan

(109, 'Sofia',  'Gonzalez', 'sofia.gonzalez@outlook.com', '0419 012 345'), -- Campus
student; low frequency

(110, 'Arjun',  'Singh',   'arjun.singh@gmail.com',    '0420 123 456'), -- Central repeater
(coffee)
```

(111, 'Zara', 'Haddad', 'zara.haddad@gmail.com', '0421 234 567'), -- Harbour dessert buyer

(112, 'Ben', 'Wilson', 'ben.wilson@company.com', '0422 345 678'), -- delivery-only occasional

(113, 'Isabella','Lopez', 'isabella.lopez@gmail.com', '0423 111 111'), -- Central brunch group; repeat

(114, 'Jack', 'O'Connor', 'jack.oconnor@outlook.com', '0424 222 222'), -- one-time Harbour tourist

(115, 'Yuki', 'Tanaka', 'yuki.tanaka@gmail.com', '0425 333 333'), -- student at Campus; cheap orders

(116, 'Carlos', 'Martinez', 'carlos.martinez@gmail.com', '0426 444 444'), -- Harbour; repeat desserts

(117, 'Amira', 'Ali', 'amira.ali@outlook.com', '0427 555 555'), -- Central; reservation & repeat

(118, 'Daniel', 'Green', 'dan.green@gmail.com', '0428 666 666'), -- Harbour; cold drinks only

(119, 'Priya', 'Sharma', 'priya.sharma@gmail.com', '0429 777 777'), -- Central; takeaway loyalist

(120, 'Mohammed','Saleh', 'm.saleh@company.com', '0430 888 888'), -- delivery only; mixed outlets

(121, 'Ella', 'Nguyen', 'ella.nguyen@gmail.com', '0431 999 999'), -- Campus; rarely orders

(122, 'George', 'Patel', 'george.patel@outlook.com', '0432 101 010'), -- Central; high-value family orders

(123, 'Chloe', 'Anderson', 'chloe.anderson@gmail.com', '0433 202 020'), -- Harbour + reservations

(124, 'Mateo', 'Silva', 'mateo.silva@gmail.com', '0434 303 030'), -- Central coffee repeat

(125, 'Hannah', 'Kim', 'hannah.kim@outlook.com', '0435 404 040'), -- Harbour; churn risk

(126, 'Oliver', 'Wright', 'oliver.wright@gmail.com', '0436 505 050'), -- Central; repeat latte buyer

(127, 'Emily', 'Davis', 'emily.davis@gmail.com', '0437 606 060'), -- Campus; only 1 order

(128, 'Jacob', 'Foster', 'jacob.foster@company.com', '0438 707 070'), -- Central + Harbour orders

(129, 'Sara', 'Hassan', 'sara.hassan@gmail.com', '0439 808 080'), -- Harbour; loyal customer

(130, 'Leo', 'Baker', 'leo.baker@gmail.com', '0440 909 090'); -- Central + delivery

Inserting Dummy Data into PRODUCT_CATEGORY Table

INSERT INTO Product_Category (Product_Category_ID, Product_Category_Name, Product_Category_Description) VALUES

(1, 'Espresso Classics', 'Short black, long black, doppio'),

(2, 'Milk Coffees', 'Latte, cappuccino, flat white'),

(3, 'Specialty Coffees', 'Mocha and flavoured coffee drinks'),

(4, 'Teas', 'Black, green, chai, herbal'),

(5, 'Hot Chocolate', 'Cocoa-based hot drinks'),

(6, 'Iced Coffees', 'Iced latte, iced mocha'),

(7, 'Cold Brew', 'Slow-brewed chilled coffees'),

(8, 'Smoothies', 'Fruit and yoghurt smoothies'),

(9, 'Fresh Juices', 'Pressed and blended juices'),

(10, 'Breakfast Meals', 'Eggs, toasties, pancakes, granola'),

(11, 'Sandwiches & Wraps', 'Panini, baguettes, wraps'),

(12, 'Salads & Bowls', 'Healthy bowls and salad plates'),

(13, 'Cakes & Pastries', 'Cakes, croissants, tarts'),

(14, 'Cookies & Brownies', 'Cookies, brownies and slices'),

(15, 'Seasonal Specials', 'Limited-time flavours and promotions');

Inserting Dummy Data into PRODUCT Table

```
INSERT INTO Product (Product_ID, Product_Name, Price, Category_ID) VALUES
-- Espresso Classics (Category 1)
(1001, 'Short Black',       3.50, 1),
(1002, 'Long Black',        3.80, 1),

-- Milk Coffees (Category 2)
(1003, 'Latte',             4.90, 2),
(1004, 'Cappuccino',        4.90, 2),
(1005, 'Flat White',        4.90, 2),

-- Specialty Coffees (Category 3)
(1006, 'Mocha',             5.20, 3),
(1007, 'Caramel Latte',     5.50, 3),

-- Teas (Category 4)
(1008, 'English Breakfast Tea', 4.00, 4),
(1009, 'Green Tea',         4.20, 4),

-- Hot Chocolate (Category 5)
(1010, 'Classic Hot Chocolate', 5.00, 5),

-- Iced Coffees (Category 6)
(1011, 'Iced Latte',        6.50, 6),
(1012, 'Iced Mocha',        7.00, 6),

-- Cold Brew (Category 7)
(1013, 'Cold Brew Coffee',  6.50, 7),
```

```
-- Smoothies (Category 8)
(1014, 'Mango Smoothie',      7.50, 8),
(1015, 'Berry Smoothie',      7.50, 8),

-- Fresh Juices (Category 9)
(1016, 'Orange Juice',        6.00, 9),
(1017, 'Apple Juice',         6.00, 9),

-- Breakfast Meals (Category 10)
(1018, 'Avocado Toast',       10.50, 10),
(1019, 'Pancakes Stack',      12.00, 10),

-- Sandwiches & Wraps (Category 11)
(1020, 'Chicken Wrap',        11.00, 11),
(1021, 'Veggie Panini',       10.50, 11),

-- Salads & Bowls (Category 12)
(1022, 'Caesar Salad',        13.50, 12),
(1023, 'Quinoa Bowl',         14.00, 12),

-- Cakes & Pastries (Category 13)
(1024, 'Cheesecake Slice',    6.50, 13),
(1025, 'Croissant',           4.50, 13),

-- Cookies & Brownies (Category 14)
(1026, 'Chocolate Brownie',   5.00, 14),
(1027, 'Oatmeal Cookie',      3.80, 14),
```

```
-- Seasonal Specials (Category 15)
(1028, 'Pumpkin Spice Latte',  6.80, 15),
(1029, 'Gingerbread Latte',    6.80, 15),
(1030, 'Summer Berry Frappe',  7.20, 15);

# Inserting Dummy data into SHIFT Table

INSERT INTO Shift (Shift_ID, Outlet_ID, Start_Time, End_Time, Max_Staff) VALUES
-- Central Outlet (ID = 1) → busiest, high Max_Staff (4–6)
(1001, 1, '06:00', '10:00', 5),
(1002, 1, '10:00', '14:00', 6),
(1003, 1, '14:00', '18:00', 6),
(1004, 1, '18:00', '22:00', 5),
(1005, 1, '07:00', '11:00', 5),
(1006, 1, '11:00', '15:00', 6),
(1007, 1, '15:00', '19:00', 5),
(1008, 1, '19:00', '23:00', 4),
(1009, 1, '08:00', '12:00', 6),
(1010, 1, '12:00', '16:00', 6),

-- Harbour Outlet (ID = 2) → moderate demand, Max_Staff (3–5)
(2001, 2, '07:00', '11:00', 4),
(2002, 2, '11:00', '15:00', 5),
(2003, 2, '15:00', '19:00', 4),
(2004, 2, '19:00', '23:00', 3),
(2005, 2, '08:00', '12:00', 4),
(2006, 2, '12:00', '16:00', 4),
```

(2007, 2, '16:00', '20:00', 5),

(2008, 2, '20:00', '00:00', 3),

(2009, 2, '09:00', '13:00', 5),

(2010, 2, '13:00', '17:00', 4),

-- Campus Outlet (ID = 3) → least busy, often under-staffed (Max_Staff 2–3)

(3001, 3, '07:00', '11:00', 3),

(3002, 3, '11:00', '15:00', 3),

(3003, 3, '15:00', '19:00', 2),

(3004, 3, '09:00', '13:00', 3),

(3005, 3, '13:00', '17:00', 2),

(3006, 3, '10:00', '14:00', 3),

(3007, 3, '14:00', '18:00', 2),

(3008, 3, '08:00', '12:00', 3),

(3009, 3, '12:00', '16:00', 2),

(3010, 3, '16:00', '20:00', 2);

Inserting Dummy Data into RESERVATION Table

INSERT INTO Reservation (Reservation_ID, Outlet_ID, Customer_ID, Date, Time, Number_of_People, Status) VALUES

-- Central Outlet (ID = 1) – busiest, mix of completed, booked, repeaters

(5001, 1, 101, '2025-09-20', '12:30', 2, 'Completed'),

(5002, 1, 101, '2025-09-27', '18:30', 3, 'Completed'),

(5003, 1, 106, '2025-09-21', '09:00', 1, 'Completed'),

(5004, 1, 110, '2025-09-22', '14:00', 2, 'Booked'),

(5005, 1, 117, '2025-09-23', '19:00', 4, 'Completed'),

(5006, 1, 122, '2025-09-24', '11:00', 5, 'Completed'),

```
(5007, 1, 124, '2025-09-25', '10:00', 1, 'Cancelled'),
(5008, 1, 126, '2025-09-26', '08:30', 2, 'Completed'),
(5009, 1, 128, '2025-09-27', '13:30', 2, 'Completed'),
(5010, 1, 130, '2025-09-28', '12:00', 3, 'Booked'),

-- Harbour Outlet (ID = 2) – more tourists, cold drinks/desserts, some no-shows
(5011, 2, 103, '2025-09-20', '18:00', 2, 'Completed'),
(5012, 2, 104, '2025-09-21', '19:00', 4, 'Completed'),
(5013, 2, 108, '2025-09-22', '20:00', 3, 'Completed'),
(5014, 2, 111, '2025-09-23', '15:00', 2, 'Completed'),
(5015, 2, 114, '2025-09-24', '13:00', 2, 'No-show'),
(5016, 2, 116, '2025-09-25', '17:00', 2, 'Completed'),
(5017, 2, 118, '2025-09-26', '14:00', 1, 'Completed'),
(5018, 2, 123, '2025-09-27', '19:00', 5, 'Completed'),
(5019, 2, 125, '2025-09-28', '11:00', 2, 'Cancelled'),
(5020, 2, 129, '2025-09-29', '12:30', 3, 'Completed'),

-- Campus Outlet (ID = 3) – fewer bookings, more no-shows/cancellations
(5021, 3, 105, '2025-09-20', '10:00', 2, 'No-show'),
(5022, 3, 109, '2025-09-21', '09:30', 1, 'Completed'),
(5023, 3, 115, '2025-09-22', '12:00', 2, 'Cancelled'),
(5024, 3, 121, '2025-09-23', '11:30', 1, 'Completed'),
(5025, 3, 127, '2025-09-24', '14:00', 2, 'No-show'),
(5026, 3, 121, '2025-09-25', '10:00', 1, 'Booked'),
(5027, 3, 105, '2025-09-26', '12:30', 2, 'Cancelled'),
(5028, 3, 109, '2025-09-27', '15:00', 2, 'Completed'),
(5029, 3, 115, '2025-09-28', '13:30', 3, 'No-show'),
(5030, 3, 127, '2025-09-29', '16:00', 2, 'Completed');
```

Inserting Dummy Data into ORDER Table

INSERT INTO Orders (Order_ID, Outlet_ID, Customer_ID, Order_Date, Order_Type, Order_Status) VALUES

-- Central Outlet (ID = 1) – busiest, mix of dine-in/takeaway

(9001, 1, 101, '2025-09-20', 'Dine-in', 'Completed'),

(9002, 1, 101, '2025-09-27', 'Dine-in', 'Completed'),

(9003, 1, 106, '2025-09-21', 'Takeaway', 'Completed'),

(9004, 1, 110, '2025-09-22', 'Dine-in', 'Pending'),

(9005, 1, 117, '2025-09-23', 'Dine-in', 'Completed'),

(9006, 1, 122, '2025-09-24', 'Dine-in', 'Completed'),

(9007, 1, 124, '2025-09-25', 'Takeaway', 'Cancelled'),

(9008, 1, 126, '2025-09-26', 'Dine-in', 'Completed'),

(9009, 1, 128, '2025-09-27', 'Delivery', 'Completed'),

(9010, 1, 130, '2025-09-28', 'Dine-in', 'Pending'),

-- Harbour Outlet (ID = 2) – strong desserts/cold drinks, tourists, evening bias

(9011, 2, 103, '2025-09-20', 'Delivery', 'Completed'),

(9012, 2, 104, '2025-09-21', 'Dine-in', 'Completed'),

(9013, 2, 108, '2025-09-22', 'Dine-in', 'Completed'),

(9014, 2, 111, '2025-09-23', 'Dine-in', 'Completed'),

(9015, 2, 114, '2025-09-24', 'Dine-in', 'No-show'),

(9016, 2, 116, '2025-09-25', 'Dine-in', 'Completed'),

(9017, 2, 118, '2025-09-26', 'Takeaway', 'Completed'),

(9018, 2, 123, '2025-09-27', 'Dine-in', 'Completed'),

(9019, 2, 125, '2025-09-28', 'Dine-in', 'Cancelled'),

(9020, 2, 129, '2025-09-29', 'Delivery', 'Completed'),

-- Campus Outlet (ID = 3) – fewer, small size orders, high cancellation rate

(9021, 3, 105, '2025-09-20', 'Dine-in', 'No-show'),

(9022, 3, 109, '2025-09-21', 'Takeaway', 'Completed'),

(9023, 3, 115, '2025-09-22', 'Dine-in', 'Cancelled'),

(9024, 3, 121, '2025-09-23', 'Dine-in', 'Completed'),

(9025, 3, 127, '2025-09-24', 'Takeaway', 'No-show'),

(9026, 3, 121, '2025-09-25', 'Dine-in', 'Pending'),

(9027, 3, 105, '2025-09-26', 'Takeaway', 'Cancelled'),

(9028, 3, 109, '2025-09-27', 'Dine-in', 'Completed'),

(9029, 3, 115, '2025-09-28', 'Delivery', 'No-show'),

(9030, 3, 127, '2025-09-29', 'Dine-in', 'Completed');

Inserting Dummy Data into PAYMENT Table

INSERT INTO Payment (Payment_ID, Order_ID, Payment_Amount, Payment_Method, Payment_Status) VALUES

-- Central Outlet (9001–9010)

(7001, 9001, 25.80, 'Card', 'Paid'),

(7002, 9002, 32.40, 'Cash', 'Paid'),

(7003, 9003, 12.90, 'Card', 'Paid'),

(7004, 9004, 15.60, 'Card', 'Pending'),

(7005, 9005, 45.00, 'Online', 'Paid'),

(7006, 9006, 52.20, 'Card', 'Paid'),

(7007, 9007, 18.50, 'Cash', 'Refunded'), -- cancelled order

(7008, 9008, 20.00, 'Card', 'Paid'),

(7009, 9009, 28.70, 'Online', 'Paid'),

(7010, 9010, 17.90, 'Card', 'Pending'),

```
-- Harbour Outlet (9011–9020)
(7011, 9011, 22.50, 'Online', 'Paid'),
(7012, 9012, 38.20, 'Card',   'Paid'),
(7013, 9013, 30.00, 'Cash',   'Paid'),
(7014, 9014, 27.50, 'Card',   'Paid'),
(7015, 9015, 0.00,  'Card',   'Failed'),    -- no-show
(7016, 9016, 40.00, 'Card',   'Paid'),
(7017, 9017, 12.00, 'Cash',   'Paid'),
(7018, 9018, 55.80, 'Online', 'Paid'),
(7019, 9019, 0.00,  'Card',   'Refunded'),  -- cancelled
(7020, 9020, 23.40, 'Card',   'Paid'),

-- Campus Outlet (9021–9030)
(7021, 9021, 0.00,  'Cash',   'Failed'),    -- no-show
(7022, 9022, 8.90,  'Card',   'Paid'),
(7023, 9023, 0.00,  'Online', 'Refunded'),  -- cancelled
(7024, 9024, 6.50,  'Cash',   'Paid'),
(7025, 9025, 0.00,  'Card',   'Failed'),    -- takeaway no-show
(7026, 9026, 10.20, 'Card',   'Pending'),
(7027, 9027, 0.00,  'Cash',   'Refunded'),  -- cancelled
(7028, 9028, 14.60, 'Online', 'Paid'),
(7029, 9029, 0.00,  'Card',   'Failed'),    -- delivery no-show
(7030, 9030, 18.40, 'Cash',   'Paid');

# Inserting Dummy Data into STAFF Table
```

```
INSERT INTO Staff (Staff_ID, First_Name, Last_Name, Designation, Employment_Status,
Manager_ID, Outlet_ID) VALUES
-- Central Outlet (ID = 1) – busiest outlet, bigger team
(201, 'Sarah',  'Lee',    'Manager',  'Active',   NULL, 1),
(202, 'John',   'Smith',  'Barista',  'Active',   201,  1),
(203, 'Emma',   'Jones',  'Server',   'Active',   201,  1),
(204, 'David',  'Brown',  'Chef',     'Active',   201,  1),
(205, 'Maya',   'Nguyen', 'Barista',  'Active',   201,  1),
(206, 'James',  'Wilson', 'Server',   'On Leave', 201,  1),
(207, 'Olivia', 'Taylor', 'Chef',     'Active',   201,  1),
(208, 'Noah',   'White',  'Server',   'Resigned', 201,  1),
(209, 'Grace',  'Martin', 'Barista',  'Active',   201,  1),
(210, 'Leo',    'Thomas', 'Server',   'Part-Time',201,  1),

-- Harbour Outlet (ID = 2) – balanced team, tourist-heavy
(211, 'Marco',  'Diaz',   'Manager',  'Active',   NULL, 2),
(212, 'Chloe',  'Nguyen', 'Barista',  'Active',   211,  2),
(213, 'Daniel', 'Kim',    'Server',   'Active',   211,  2),
(214, 'Sophia', 'Hernandez','Chef',   'Active',   211,  2),
(215, 'Lucas',  'Patel',  'Barista',  'Active',   211,  2),
(216, 'Aria',   'Ali',    'Server',   'On Leave', 211,  2),
(217, 'Ethan',  'Wong',   'Chef',     'Active',   211,  2),
(218, 'Mila',   'Singh',  'Server',   'Resigned', 211,  2),
(219, 'Liam',   'Garcia', 'Barista',  'Active',   211,  2),
(220, 'Isla',   'Rodriguez','Server',  'Part-Time',211,  2),

-- Campus Outlet (ID = 3) – smaller, under-staffed
(221, 'Henry',  'Park',   'Manager',  'Active',   NULL, 3),
```

(222, 'Zoe', 'Lopez', 'Barista', 'Active', 221, 3),

(223, 'Ryan', 'Khan', 'Server', 'Active', 221, 3),

(224, 'Lily', 'Adams', 'Chef', 'On Leave', 221, 3),

(225, 'Eli', 'Sharma', 'Barista', 'Active', 221, 3),

(226, 'Ava', 'Baker', 'Server', 'Active', 221, 3),

(227, 'Omar', 'Davies', 'Chef', 'Resigned', 221, 3),

(228, 'Ella', 'Ng', 'Server', 'Part-Time',221, 3),

(229, 'Adam', 'Hassan', 'Barista', 'Active', 221, 3),

(230, 'Nora', 'Scott', 'Server', 'Active', 221, 3);

Inserting Dummy Data into OUTLET-PRODUCT Table

INSERT INTO Outlet_Product (Product_ID, Outlet_ID, Outlet_Status) VALUES

-- Central (ID = 1) → wide menu, everything available

(1001, 1, 'Available'),

(1002, 1, 'Available'),

(1003, 1, 'Available'),

(1004, 1, 'Available'),

(1005, 1, 'Available'),

(1006, 1, 'Available'),

(1018, 1, 'Available'),

(1020, 1, 'Available'),

(1022, 1, 'Available'),

(1024, 1, 'Available'),

-- Harbour (ID = 2) → strong in Cold Drinks/Desserts, some meals missing

(1011, 2, 'Available'),

```
(1012, 2, 'Available'),
(1013, 2, 'Unavailable'),   -- Cold Brew not offered
(1014, 2, 'Available'),
(1016, 2, 'Available'),
(1024, 2, 'Available'),
(1025, 2, 'Available'),
(1026, 2, 'Available'),
(1019, 2, 'Unavailable'),   -- Pancakes not served
(1023, 2, 'Available'),

-- Campus (ID = 3) → limited to Beverages & Desserts
(1001, 3, 'Available'),
(1003, 3, 'Available'),
(1004, 3, 'Available'),
(1010, 3, 'Available'),
(1011, 3, 'Unavailable'),   -- Iced Latte not sold
(1018, 3, 'Unavailable'),   -- Meals not sold at campus
(1020, 3, 'Unavailable'),   -- Wraps not offered
(1022, 3, 'Unavailable'),   -- Salads not offered
(1024, 3, 'Available'),
(1025, 3, 'Available');



# Inserting Dummy Data into ORDER_PRODUCT Table

INSERT INTO Order_Product (Order_ID, Product_ID, Quantity, Price) VALUES
-- Central Orders
(9001, 1003, 2, 4.90),   -- 2 Lattes
```

(9001, 1024, 1, 6.50), -- Cheesecake

(9002, 1004, 3, 4.90), -- 3 Cappuccinos

(9002, 1020, 1, 11.00), -- Chicken Wrap

(9003, 1005, 2, 4.90), -- Flat White

(9004, 1010, 2, 5.00), -- Hot Chocolate

(9005, 1018, 2, 10.50), -- Avocado Toast

(9005, 1023, 1, 14.00), -- Quinoa Bowl

(9006, 1019, 2, 12.00), -- Pancakes

(9006, 1022, 1, 13.50), -- Caesar Salad

-- Harbour Orders

(9011, 1011, 2, 6.50), -- Iced Latte

(9012, 1014, 2, 7.50), -- Mango Smoothie

(9012, 1025, 1, 4.50), -- Croissant

(9013, 1016, 2, 6.00), -- Orange Juice

(9014, 1026, 2, 5.00), -- Brownies

(9015, 1012, 1, 7.00), -- Iced Mocha (no-show, failed)

(9016, 1023, 2, 14.00), -- Quinoa Bowl

(9017, 1017, 1, 6.00), -- Apple Juice

(9018, 1024, 2, 6.50), -- Cheesecake

(9018, 1026, 1, 5.00), -- Brownie

-- Campus Orders

(9021, 1003, 1, 4.90), -- Latte (no-show)

(9022, 1001, 1, 3.50), -- Short Black

(9022, 1025, 1, 4.50), -- Croissant

(9023, 1018, 1, 10.50), -- Avocado Toast (cancelled)

(9024, 1004, 1, 4.90), -- Cappuccino

(9025, 1003, 1, 4.90), -- Latte (no-show)

(9026, 1010, 2, 5.00), -- Hot Chocolate

(9027, 1005, 1, 4.90), -- Flat White (cancelled)

(9028, 1024, 1, 6.50), -- Cheesecake

(9030, 1003, 2, 4.90); -- Lattes

Inserting Dummy data into SHIFT-STAFF Table

INSERT INTO Staff_Shift (Staff_ID, Shift_ID, Staff_Role, Working_Hours) VALUES

-- Central Outlet (Staff 201–210, Shifts 1001–1010)

(201, 1001, 'Manager on Duty', 4.00),

(202, 1001, 'Barista', 4.00),

(203, 1002, 'Server', 4.00),

(204, 1002, 'Chef', 5.00),

(205, 1003, 'Barista', 4.00),

(207, 1004, 'Chef', 5.00),

(209, 1005, 'Barista', 4.00),

(210, 1006, 'Server', 3.50),

(206, 1007, 'Server', 0.00), -- On Leave (under-utilisation)

(208, 1008, 'Server', 0.00), -- Resigned (gap shown)

-- Harbour Outlet (Staff 211–220, Shifts 2001–2010)

(211, 2001, 'Manager on Duty', 4.00),

(212, 2001, 'Barista', 4.00),

(213, 2002, 'Server', 4.00),

(214, 2002, 'Chef', 5.00),

(215, 2003, 'Barista', 4.00),

```
(217, 2004, 'Chef',          5.00),
(219, 2005, 'Barista',       4.00),
(220, 2006, 'Server',        3.50),
(216, 2007, 'Server',        0.00),   -- On Leave
(218, 2008, 'Server',        0.00),   -- Resigned

-- Campus Outlet (Staff 221–230, Shifts 3001–3010)
(221, 3001, 'Manager on Duty', 4.00),
(222, 3001, 'Barista',       4.00),
(223, 3002, 'Server',        4.00),
(225, 3002, 'Barista',       4.00),
(226, 3003, 'Server',        3.50),
(229, 3004, 'Barista',       4.00),
(230, 3005, 'Server',        4.00),
(224, 3006, 'Chef',          0.00),   -- On Leave
(227, 3007, 'Chef',          0.00),   -- Resigned
(228, 3008, 'Server',        2.00);   -- Part-Time
```

Part B – Business Analytics Report

1.7 Business concern 1 Sales & Profitability

Name and Description: The objective is to evaluate which product categories and outlets contribute most to overall revenue and whether sales distribution aligns with operational efficiency. By analysing revenue by category, outlet order status, total outlet revenue, and category revenue per outlet, we can identify profitability drivers and leakage points.

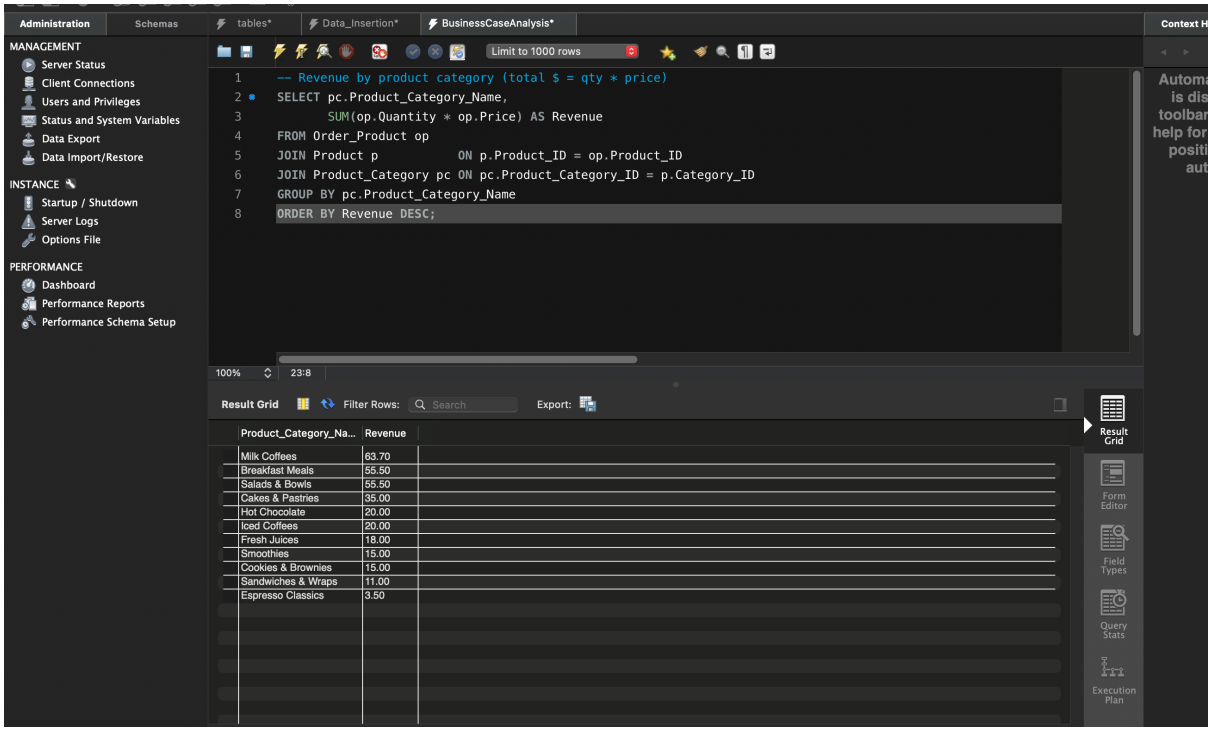
SQL Query

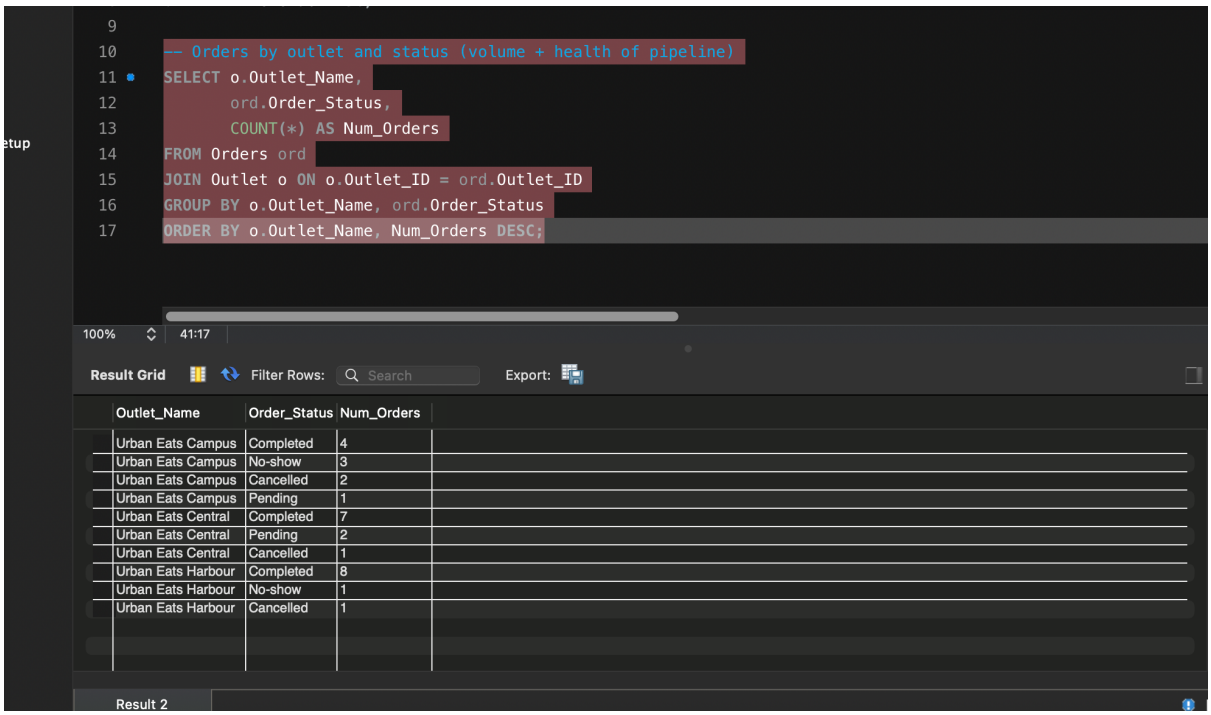
```
-- Revenue by product category (total $ = qty * price)
SELECT pc.Product_Category_Name,
    SUM(op.Quantity * op.Price) AS Revenue
FROM Order_Product op
JOIN Product p        ON p.Product_ID = op.Product_ID
JOIN Product_Category pc ON pc.Product_Category_ID = p.Category_ID
GROUP BY pc.Product_Category_Name
ORDER BY Revenue DESC;



-- Orders by outlet and status (volume + health of pipeline)
SELECT o.Outlet_Name,
    ord.Order_Status,
    COUNT(*) AS Num_Orders
FROM Orders ord
JOIN Outlet o ON o.Outlet_ID = ord.Outlet_ID
GROUP BY o.Outlet_Name, ord.Order_Status
ORDER BY o.Outlet_Name, Num_Orders DESC;

-- Total revenue by outlet (paid amounts)
SELECT o.Outlet_Name,
```

```
    SUM(p.Payment_Amount) AS Total_Revenue
FROM Payment p
JOIN Orders ord ON p.Order_ID = ord.Order_ID
JOIN Outlet o   ON ord.Outlet_ID = o.Outlet_ID
GROUP BY o.Outlet_Name
ORDER BY Total_Revenue DESC;


-- Category revenue by outlet (find which branches carry which profit buckets)
SELECT o.Outlet_Name,
    pc.Product_Category_Name,
    SUM(op.Quantity * op.Price) AS Revenue
FROM Orders ord
JOIN Outlet o         ON o.Outlet_ID = ord.Outlet_ID
JOIN Order_Product op   ON op.Order_ID = ord.Order_ID
JOIN Product p        ON p.Product_ID = op.Product_ID
JOIN Product_Category pc ON pc.Product_Category_ID = p.Category_ID
GROUP BY o.Outlet_Name, pc.Product_Category_Name
ORDER BY o.Outlet_Name, Revenue DESC;
```

SQL Result

Classified


Administration
Schemas
tables*
Data_Insertion*
BusinessCaseAnalysis*
Limit to 1000 rows
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Performance Reports
Performance Schema Setup
1 -- Revenue by product category (total $ = qty * price)
2 SELECT pc.Product_Category_Name,
3 SUM(op.Quantity * op.Price) AS Revenue
4 FROM Order_Product op
5 JOIN Product p ON p.Product_ID = op.Product_ID
6 JOIN Product_Category pc ON pc.Product_Category_ID = p.Category_ID
7 GROUP BY pc.Product_Category_Name
8 ORDER BY Revenue DESC;
100%
23:8
Result Grid
Filter Rows:
Search
Export:
Product_Category_Na...
Revenue
Milk Coffees 63.70
Breakfast Meals 55.50
Salads & Bowls 55.50
Cakes & Pastries 35.00
Hot Chocolate 20.00
Iced Coffees 20.00
Fresh Juices 18.00
Smoothies 15.00
Cookies & Brownies 15.00
Sandwiches & Wraps 11.00
Espresso Classics 3.50
Result Grid
Form Editor
Field Types
Query Stats
Execution Plan


9
10 -- Orders by outlet and status (volume + health of pipeline)
11 SELECT o.Outlet_Name,
12 ord.Order_Status,
13 COUNT(*) AS Num_Orders
14 FROM Orders ord
15 JOIN Outlet o ON o.Outlet_ID = ord.Outlet_ID
16 GROUP BY o.Outlet_Name, ord.Order_Status
17 ORDER BY o.Outlet_Name, Num_Orders DESC;
100%
41:17
Result Grid
Filter Rows:
Search
Export:
Outlet_Name
Order_Status
Num_Orders
Urban Eats Campus Completed 4
Urban Eats Campus No-show 3
Urban Eats Campus Cancelled 2
Urban Eats Campus Pending 1
Urban Eats Central Completed 7
Urban Eats Central Pending 2
Urban Eats Central Cancelled 1
Urban Eats Harbour Completed 8
Urban Eats Harbour No-show 1
Urban Eats Harbour Cancelled 1
Result 2

```sql
-- Total revenue by outlet (paid amounts)
SELECT o.Outlet_Name,
       SUM(p.Payment_Amount) AS Total_Revenue
FROM Payment p
JOIN Orders ord ON p.Order_ID = ord.Order_ID
JOIN Outlet o   ON ord.Outlet_ID = o.Outlet_ID
GROUP BY o.Outlet_Name
ORDER BY Total_Revenue DESC;
```

00% 23:25

Result Grid Filter Rows: Search Export:

Outlet_Name	Total_Revenue
Urban Eats Central	269.00
Urban Eats Harbour	249.40
Urban Eats Campus	58.60

```sql
-- Category revenue by outlet (find which branches carry which profit buckets)
SELECT o.Outlet_Name,
       pc.Product_Category_Name,
       SUM(op.Quantity * op.Price) AS Revenue
FROM Orders ord
JOIN Outlet o           ON o.Outlet_ID = ord.Outlet_ID
JOIN Order_Product op   ON op.Order_ID = ord.Order_ID
JOIN Product p          ON p.Product_ID = op.Product_ID
JOIN Product_Category pc ON pc.Product_Category_ID = p.Category_ID
GROUP BY o.Outlet_Name, pc.Product_Category_Name
ORDER BY o.Outlet_Name, Revenue DESC;
```

100% 38:38

Result Grid Filter Rows: Search Export:

Outlet_Name	Product_Category_Na...	Revenue
Urban Eats Campus	Milk Coffees	29.40
Urban Eats Campus	Cakes & Pastries	11.00
Urban Eats Campus	Breakfast Meals	10.50
Urban Eats Campus	Hot Chocolate	10.00
Urban Eats Campus	Espresso Classics	3.50
Urban Eats Central	Breakfast Meals	45.00
Urban Eats Central	Milk Coffees	34.30
Urban Eats Central	Salads & Bowls	27.50
Urban Eats Central	Sandwiches & Wraps	11.00
Urban Eats Central	Hot Chocolate	10.00
Urban Eats Central	Cakes & Pastries	6.50
Urban Eats Harbour	Salads & Bowls	28.00
Urban Eats Harbour	Iced Coffees	20.00
Urban Eats Harbour	Fresh Juices	18.00
Urban Eats Harbour	Cakes & Pastries	17.50
Urban Eats Harbour	Smoothies	15.00
Urban Eats Harbour	Cookies & Brownies	15.00

Result 4 Read

Justification and Insights

Central is the most profitable outlet with a balanced category mix, sustaining consistent revenue streams across beverages, meals, and desserts. Harbour thrives on its cold-drink and dessert niche, performing strongly despite limited high-margin meal offerings. Campus faces severe profitability challenges, generating only a fraction of the revenue

of other outlets due to cancellations, no-shows, and restricted product variety. Espresso Classics should be reviewed for either rebranding or removal, as they show minimal contribution. Strategic actions include expanding meal offerings at Campus to capture higher ticket sales, strengthening operational control over reservations to reduce no-shows, and rebalancing marketing focus to promote underperforming but high-margin categories.

1.8 Business concern 2 Customer Retention

Name and Description: The goal is to assess which customers show loyalty through repeat purchases, which are at risk of churn, and how reservation behaviour signals long-term retention. By linking order frequency, payment status, and reservation completion, we can identify profitable repeaters versus high-risk one-time or failed customers.

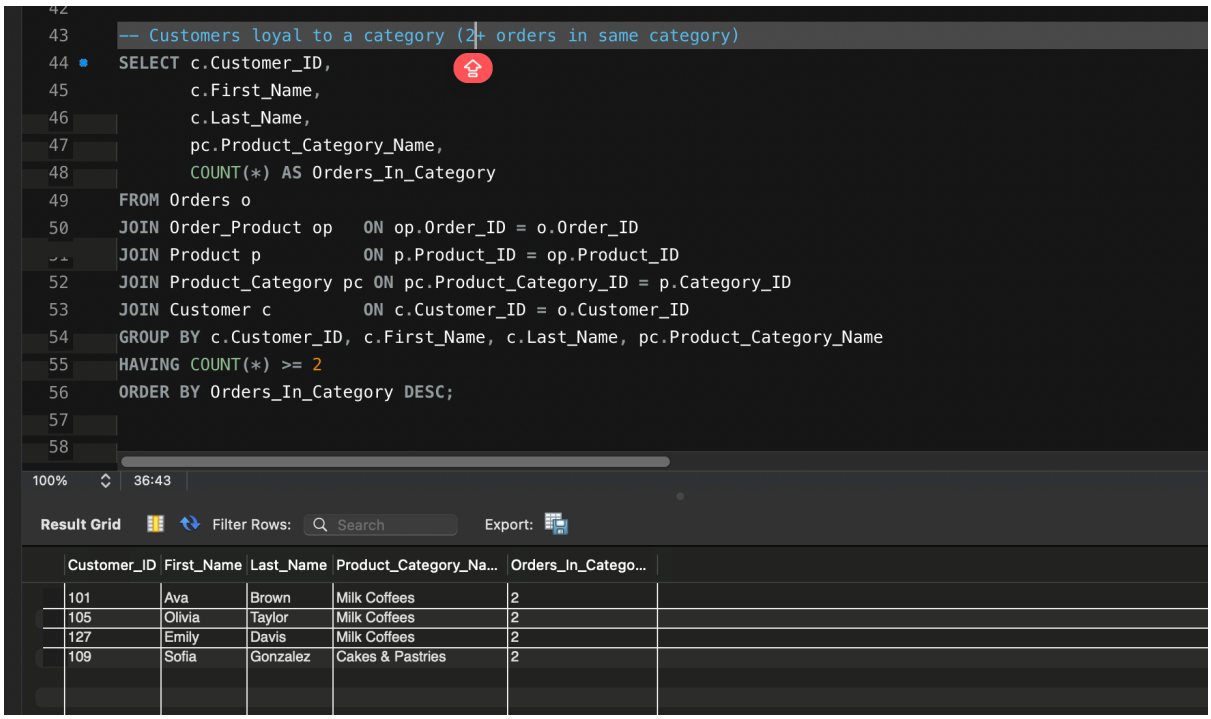
SQL Query

```
-- Customers loyal to a category (2+ orders in same category)
SELECT c.Customer_ID,
    c.First_Name,
    c.Last_Name,
    pc.Product_Category_Name,
    COUNT(*) AS Orders_In_Category
FROM Orders o
JOIN Order_Product op   ON op.Order_ID = o.Order_ID
JOIN Product p        ON p.Product_ID = op.Product_ID
JOIN Product_Category pc ON pc.Product_Category_ID = p.Category_ID
JOIN Customer c       ON c.Customer_ID = o.Customer_ID
GROUP BY c.Customer_ID, c.First_Name, c.Last_Name, pc.Product_Category_Name
HAVING COUNT(*) >= 2
ORDER BY Orders_In_Category DESC;


-- Loyal vs one-timer by outlet (customer order counts per outlet)
SELECT o.Outlet_Name,
```

```
    c.Customer_ID,
    c.First_Name,
    c.Last_Name,
    COUNT(*) AS Orders_At_Outlet,
    CASE WHEN COUNT(*) >= 2 THEN 'Loyal' ELSE 'One-timer' END AS Segment
FROM Orders ord
JOIN Outlet o   ON o.Outlet_ID = ord.Outlet_ID
JOIN Customer c ON c.Customer_ID = ord.Customer_ID
GROUP BY o.Outlet_Name, c.Customer_ID, c.First_Name, c.Last_Name
ORDER BY o.Outlet_Name, Orders_At_Outlet DESC;

-- Failed / no-show signals tied to churn risk
SELECT c.Customer_ID,
    c.First_Name,
    c.Last_Name,
    ord.Order_ID,
    COALESCE(p.Payment_Status, 'No Payment') AS Payment_Status,
    ord.Order_Status
FROM Orders ord
LEFT JOIN Payment p ON p.Order_ID = ord.Order_ID
JOIN Customer c     ON c.Customer_ID = ord.Customer_ID
WHERE COALESCE(p.Payment_Status, 'No Payment') IN
('Failed','Refunded','Pending','No Payment')
  OR ord.Order_Status IN ('Cancelled','No-show')
ORDER BY c.Customer_ID, ord.Order_ID;

-- Reservation  - order conversion rate by outlet (loyalty signal)
-- Assumes Reservation has Customer_ID and an Order can be linked via Customer +
date window or a Reservation_ID if present.
```

```
SELECT o.Outlet_Name,
    SUM(CASE WHEN r.Status = 'Completed' THEN 1 ELSE 0 END)              AS Reservations_Completed,
    SUM(CASE WHEN r.Status IN ('No-show','Cancelled') THEN 1 ELSE 0 END)  AS Reservations_Failed,
    COUNT(*)                                                             AS Reservations_Total,
    ROUND(100.0 * SUM(CASE WHEN r.Status='Completed' THEN 1 ELSE 0 END)
        / NULLIF(COUNT(*),0), 1)                                         AS Reservation_Completion_Pct
FROM Reservation r
JOIN Outlet o ON o.Outlet_ID = r.Outlet_ID
GROUP BY o.Outlet_Name
ORDER BY Reservation_Completion_Pct DESC;
```

SQL Result



Classified

```sql
-- Loyal vs one-timer by outlet (customer order counts per outlet)
SELECT o.Outlet_Name,
       c.Customer_ID,
       c.First_Name,
       c.Last_Name,
       COUNT(*) AS Orders_At_Outlet,
       CASE WHEN COUNT(*) >= 2 THEN 'Loyal' ELSE 'One-timer' END AS Segment
FROM Orders ord
JOIN Outlet o   ON o.Outlet_ID = ord.Outlet_ID
JOIN Customer c ON c.Customer_ID = ord.Customer_ID
GROUP BY o.Outlet_Name, c.Customer_ID, c.First_Name, c.Last_Name
ORDER BY o.Outlet_Name, Orders_At_Outlet DESC;
```

100% 47:70

Result Grid Filter Rows: Search Export:

Outlet_Name	Customer_ID	First_Name	Last_Name	Orders_At_Outl...	Segment
Urban Eats Central	130	Leo	Baker	1	One-timer
Urban Eats Central	124	Mateo	Silva	1	One-timer
Urban Eats Central	122	George	Patel	1	One-timer
Urban Eats Central	117	Amira	Ali	1	One-timer
Urban Eats Central	110	Arjun	Singh	1	One-timer
Urban Eats Central	106	Lucas	White	1	One-timer
Urban Eats Harbour	103	Noah	Khan	1	One-timer
Urban Eats Harbour	104	Mia	Chen	1	One-timer
Urban Eats Harbour	108	Ethan	Rao	1	One-timer
Urban Eats Harbour	114	Jack	O'Connor	1	One-timer
Urban Eats Harbour	116	Carlos	Martinez	1	One-timer
Urban Eats Harbour	118	Daniel	Green	1	One-timer
Urban Eats Harbour	123	Chloe	Anderson	1	One-timer
Urban Eats Harbour	125	Hannah	Kim	1	One-timer
Urban Eats Harbour	129	Sara	Hassan	1	One-timer
Urban Eats Harbour	111	Zara	Haddad	1	One-timer

Result Grid Form Editor Field Types Query Stats

Result 7 Read Only

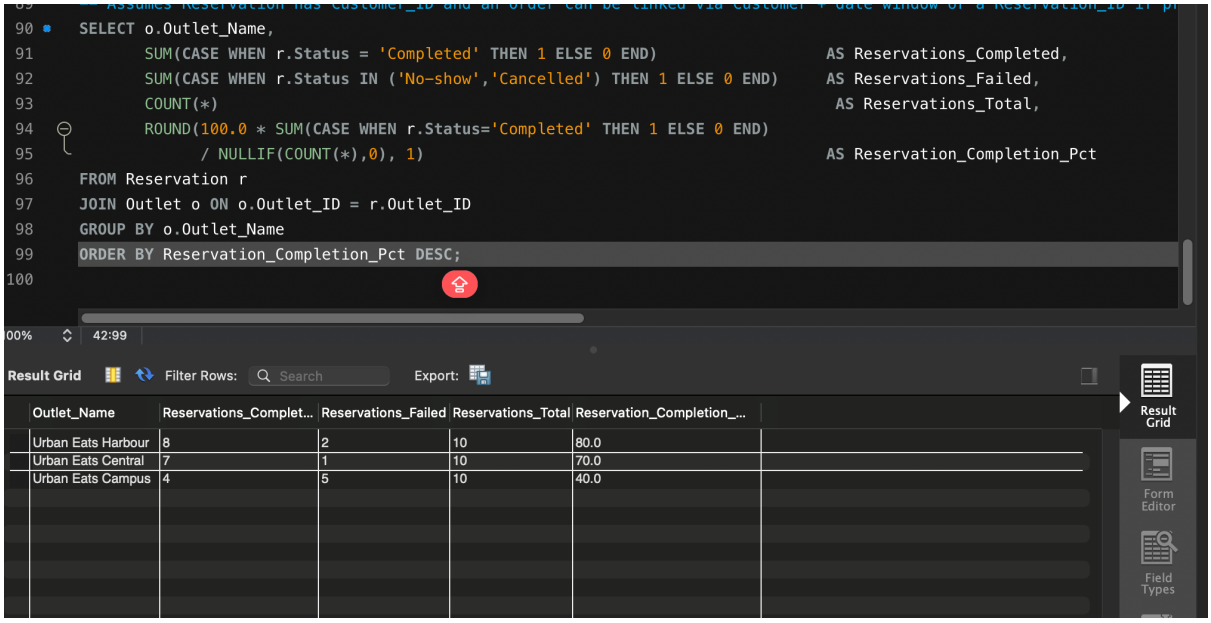
```sql
-- Failed / no-show signals tied to churn risk
SELECT c.Customer_ID,
       c.First_Name,
       c.Last_Name,
       ord.Order_ID,
       COALESCE(p.Payment_Status, 'No Payment') AS Payment_Status,
       ord.Order_Status
FROM Orders ord
LEFT JOIN Payment p ON p.Order_ID = ord.Order_ID
JOIN Customer c     ON c.Customer_ID = ord.Customer_ID
WHERE COALESCE(p.Payment_Status, 'No Payment') IN ('Failed','Refunded','Pending','No Payment')
   OR ord.Order_Status IN ('Cancelled','No-show')
ORDER BY c.Customer_ID, ord.Order_ID;
```

100% 1:86

Result Grid Filter Rows: Search Export:

Customer_ID	First_Name	Last_Name	Order_ID	Payment_Status	Order_Status
105	Olivia	Taylor	9021	Failed	No-show
105	Olivia	Taylor	9027	Refunded	Cancelled
110	Arjun	Singh	9004	Pending	Pending
114	Jack	O'Connor	9015	Failed	No-show
115	Yuki	Tanaka	9023	Refunded	Cancelled
115	Yuki	Tanaka	9029	Failed	No-show
121	Ella	Nguyen	9026	Pending	Pending
124	Mateo	Silva	9007	Refunded	Cancelled
125	Hannah	Kim	9019	Refunded	Cancelled
127	Emily	Davis	9025	Failed	No-show
130	Leo	Baker	9010	Pending	Pending

Result Grid Form Editor Field Types Query Stats



```
90  SELECT o.Outlet_Name,
91         SUM(CASE WHEN r.Status = 'Completed' THEN 1 ELSE 0 END)                AS Reservations_Completed,
92         SUM(CASE WHEN r.Status IN ('No-show','Cancelled') THEN 1 ELSE 0 END)   AS Reservations_Failed,
93         COUNT(*)                                                               AS Reservations_Total,
94         ROUND(100.0 * SUM(CASE WHEN r.Status='Completed' THEN 1 ELSE 0 END)
95               / NULLIF(COUNT(*),0), 1)                                         AS Reservation_Completion_Pct
96  FROM Reservation r
97  JOIN Outlet o ON o.Outlet_ID = r.Outlet_ID
98  GROUP BY o.Outlet_Name
99  ORDER BY Reservation_Completion_Pct DESC;
100
```

100% 42:99

Result Grid | Filter Rows: Search | Export:

Outlet_Name	Reservations_Complet...	Reservations_Failed	Reservations_Total	Reservation_Completion_...
Urban Eats Harbour	8	2	10	80.0
Urban Eats Central	7	1	10	70.0
Urban Eats Campus	4	5	10	40.0

Result Grid | Form Editor | Field Types

Justification and Insights

Loyalty exists but is fragile, with most customers concentrated in a single product category. Failed or refunded payments directly overlap with no-shows, amplifying churn risk. Harbour performs best in converting reservations to actual orders, while Campus demonstrates severe reliability issues, damaging long-term retention. Strategic actions include building loyalty incentives for category-driven customers such as beverage and dessert bundles, strengthening payment and reservation enforcement to reduce attrition, and focusing on Campus process improvements through confirmation systems and a deposit policy to boost conversion rates and retain customers.

1.9 Business concern 3 Menu Optimisation

Name and Description: This analysis identifies which product categories and individual menu items are underperforming, and how sales distribution varies by outlet. The objective is to optimise the menu by pruning weak items, promoting consistent performers, and aligning product availability with outlet demand.

SQL Query

```
-- Underperforming categories (low order count)
SELECT pc.Product_Category_Name,
    COUNT(*) AS Times_Ordered
FROM Order_Product op
JOIN Product p        ON p.Product_ID = op.Product_ID
```

```
JOIN Product_Category pc ON pc.Product_Category_ID = p.Category_ID
GROUP BY pc.Product_Category_Name
ORDER BY Times_Ordered ASC;

-- Underperforming products (identify items to prune)
SELECT p.Product_Name,
    COUNT(*) AS Times_Ordered
FROM Order_Product op
JOIN Product p ON p.Product_ID = op.Product_ID
GROUP BY p.Product_Name
ORDER BY Times_Ordered ASC, p.Product_Name;

-- Product sales by outlet (see where each item actually sells)
SELECT o.Outlet_Name,
    p.Product_Name,
    SUM(op.Quantity) AS Units_Sold,
    SUM(op.Quantity * op.Price) AS Revenue
FROM Orders ord
JOIN Outlet o       ON o.Outlet_ID = ord.Outlet_ID
JOIN Order_Product op ON op.Order_ID = ord.Order_ID
JOIN Product p      ON p.Product_ID = op.Product_ID
GROUP BY o.Outlet_Name, p.Product_Name
ORDER BY o.Outlet_Name, Revenue DESC;
```

SQL Result

```sql
# BUSINESS CONCERN 3 – MENU OPTIMISATION
-- Underperforming categories (low order count)
SELECT pc.Product_Category_Name,
       COUNT(*) AS Times_Ordered
FROM Order_Product op
JOIN Product p           ON p.Product_ID = op.Product_ID
JOIN Product_Category pc ON pc.Product_Category_ID = p.Category_ID
GROUP BY pc.Product_Category_Name
ORDER BY Times_Ordered ASC;
```

100% 28:110

Result Grid Filter Rows: Search Export:

Product_Category_Na...	Times_Ordered
Espresso Classics	1
Smoothies	1
Sandwiches & Wraps	1
Hot Chocolate	2
Iced Coffees	2
Fresh Juices	2
Cookies & Brownies	2
Breakfast Meals	3
Salads & Bowls	3
Cakes & Pastries	5
Milk Coffees	8

```sql
-- Underperforming products (identify items to prune)
SELECT p.Product_Name,
       COUNT(*) AS Times_Ordered
FROM Order_Product op
JOIN Product p ON p.Product_ID = op.Product_ID
GROUP BY p.Product_Name
ORDER BY Times_Ordered ASC, p.Product_Name;
```

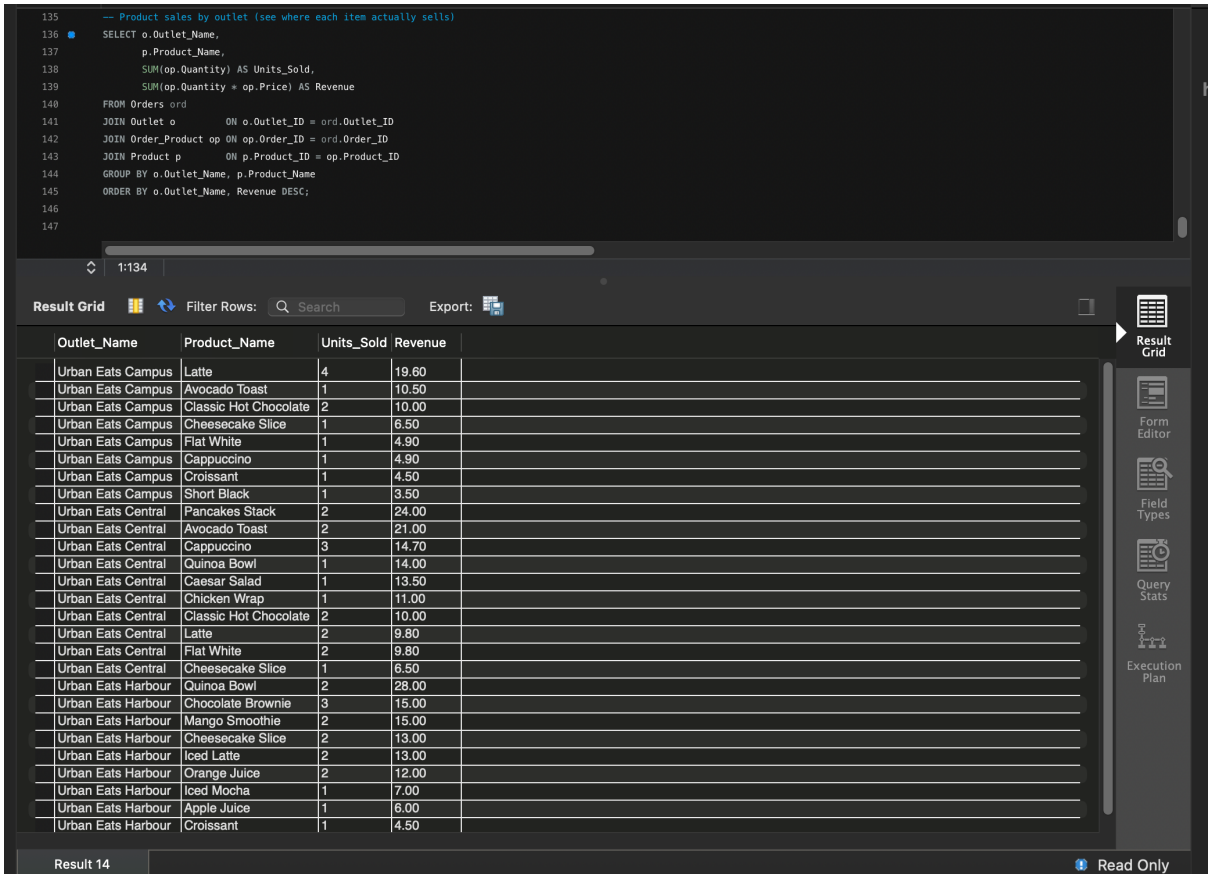
100% 1:118

Result Grid Filter Rows: Search Export:

Product_Name	Times_Ordered
Apple Juice	1
Caesar Salad	1
Chicken Wrap	1
Iced Latte	1
Iced Mocha	1
Mango Smoothie	1
Orange Juice	1
Pancakes Stack	1
Short Black	1
Avocado Toast	2
Cappuccino	2
Chocolate Brow...	2
Classic Hot Ch...	2
Croissant	2
Flat White	2
Quinoa Bowl	2
Cheesecake Slice	3
Latte	4

Result Grid Form Editor Field Types Query Stats Execution Plan

Result 11 Read Only



```
-- Product sales by outlet (see where each item actually sells)
SELECT o.Outlet_Name,
       p.Product_Name,
       SUM(op.Quantity) AS Units_Sold,
       SUM(op.Quantity * op.Price) AS Revenue
FROM Orders ord
JOIN Outlet o         ON o.Outlet_ID = ord.Outlet_ID
JOIN Order_Product op ON op.Order_ID = ord.Order_ID
JOIN Product p        ON p.Product_ID = op.Product_ID
GROUP BY o.Outlet_Name, p.Product_Name
ORDER BY o.Outlet_Name, Revenue DESC;
```

1:134

Result Grid | Filter Rows: Search | Export:

Outlet_Name	Product_Name	Units_Sold	Revenue
Urban Eats Campus	Latte	4	19.60
Urban Eats Campus	Avocado Toast	1	10.50
Urban Eats Campus	Classic Hot Chocolate	2	10.00
Urban Eats Campus	Cheesecake Slice	1	6.50
Urban Eats Campus	Flat White	1	4.90
Urban Eats Campus	Cappuccino	1	4.90
Urban Eats Campus	Croissant	1	4.50
Urban Eats Campus	Short Black	1	3.50
Urban Eats Central	Pancakes Stack	2	24.00
Urban Eats Central	Avocado Toast	2	21.00
Urban Eats Central	Cappuccino	3	14.70
Urban Eats Central	Quinoa Bowl	1	14.00
Urban Eats Central	Caesar Salad	1	13.50
Urban Eats Central	Chicken Wrap	1	11.00
Urban Eats Central	Classic Hot Chocolate	2	10.00
Urban Eats Central	Latte	2	9.80
Urban Eats Central	Flat White	2	9.80
Urban Eats Central	Cheesecake Slice	1	6.50
Urban Eats Harbour	Quinoa Bowl	2	28.00
Urban Eats Harbour	Chocolate Brownie	3	15.00
Urban Eats Harbour	Mango Smoothie	2	15.00
Urban Eats Harbour	Cheesecake Slice	2	13.00
Urban Eats Harbour	Iced Latte	2	13.00
Urban Eats Harbour	Orange Juice	2	12.00
Urban Eats Harbour	Iced Mocha	1	7.00
Urban Eats Harbour	Apple Juice	1	6.00
Urban Eats Harbour	Croissant	1	4.50

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Result 14 | Read Only

Justification and Insights

Menu has too many low-performing items with negligible demand, increasing operational costs without contributing to revenue. Campus outlet lacks diversity and relies heavily on a few products, limiting its ability to capture customer preferences. Harbour validates its tourist strategy with cold beverages and desserts driving revenue, but offers little in high-margin meals. Central is the most balanced outlet with both meals and beverages performing well. Strategic actions include pruning or rebranding consistently underperforming items such as Espresso Classics, Apple Juice, and Caesar Salad, focusing promotions on mid-tier items like Quinoa Bowls and Cappuccino that have potential but limited current traction, and reallocating menu design so that Central retains full diversity, Harbour doubles down on drinks and desserts, and Campus expands to include more meal options.

1.10 Business concern 4 Staff Scheduling and Performance

Name and Description: This analysis examines whether staff allocation is aligned with demand across outlets and shifts. It looks at utilisation rates, the ratio of active staff to total staff, and the distribution of product demand by role groups (barista, chef, server/baker). The aim is to identify inefficiencies, understaffing risks, and role mismatches that directly affect service quality and revenue.

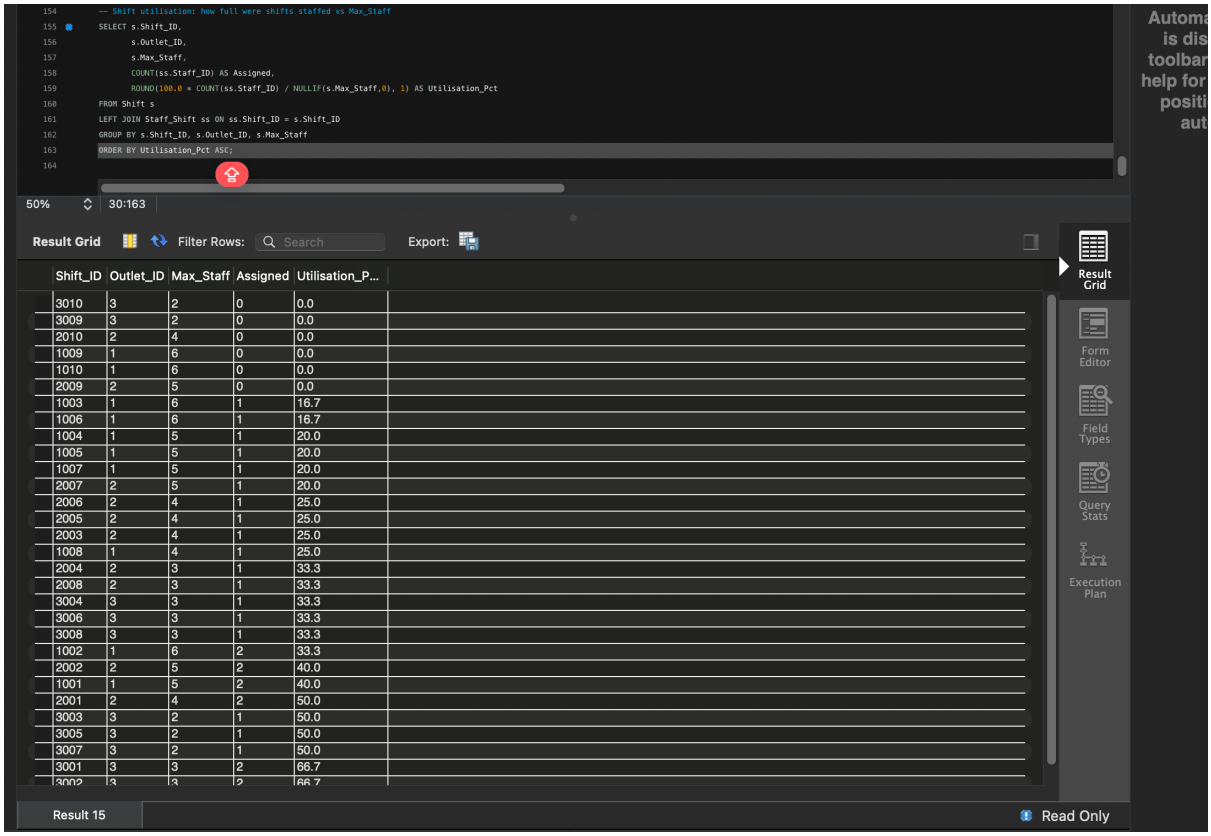
SQL Query

```
-- Shift utilisation: how full were shifts staffed vs Max_Staff
SELECT s.Shift_ID,
    s.Outlet_ID,
    s.Max_Staff,
    COUNT(ss.Staff_ID) AS Assigned,
    ROUND(100.0 * COUNT(ss.Staff_ID) / NULLIF(s.Max_Staff,0), 1) AS Utilisation_Pct
FROM Shift s
LEFT JOIN Staff_Shift ss ON ss.Shift_ID = s.Shift_ID
GROUP BY s.Shift_ID, s.Outlet_ID, s.Max_Staff
ORDER BY Utilisation_Pct ASC;

-- Active vs total staff by outlet (capacity view)
SELECT o.Outlet_Name,
    COUNT(*) AS Total_Staff,
    SUM(CASE WHEN s.Employment_Status='Active' THEN 1 ELSE 0 END) AS
Active_Staff
FROM Staff s
JOIN Outlet o ON s.Outlet_ID = o.Outlet_ID
GROUP BY o.Outlet_Name
ORDER BY Active_Staff DESC;

-- Active vs total staff by outlet (capacity view)
SELECT o.Outlet_Name,
    COUNT(*) AS Total_Staff,
    SUM(CASE WHEN s.Employment_Status='Active' THEN 1 ELSE 0 END) AS
Active_Staff
FROM Staff s
JOIN Outlet o ON s.Outlet_ID = o.Outlet_ID
```

```
GROUP BY o.Outlet_Name
ORDER BY Active_Staff DESC;


-- Category skills vs outlet demand (barista vs chef vs baker)
-- Map category groups to roles for mismatch checks.
SELECT o.Outlet_Name,
    CASE
     WHEN pc.Product_Category_Name IN ('Espresso Classics','Milk Coffees','Iced Coffees','Cold Drinks','Smoothies','Juices') THEN 'Barista'
     WHEN pc.Product_Category_Name IN ('Breakfast','Bowls & Salads','Meals') THEN 'Chef'
     WHEN pc.Product_Category_Name IN ('Cakes & Pastries','Cookies & Desserts') THEN 'Server/Baker'
     ELSE 'Other'
    END AS Role_Group,
    SUM(op.Quantity) AS Units_Sold
FROM Orders ord
JOIN Outlet o         ON o.Outlet_ID = ord.Outlet_ID
JOIN Order_Product op   ON op.Order_ID = ord.Order_ID
JOIN Product p        ON p.Product_ID = op.Product_ID
JOIN Product_Category pc ON pc.Product_Category_ID = p.Category_ID
GROUP BY o.Outlet_Name, Role_Group
ORDER BY o.Outlet_Name, Units_Sold DESC;
```

SQL Result

Classified



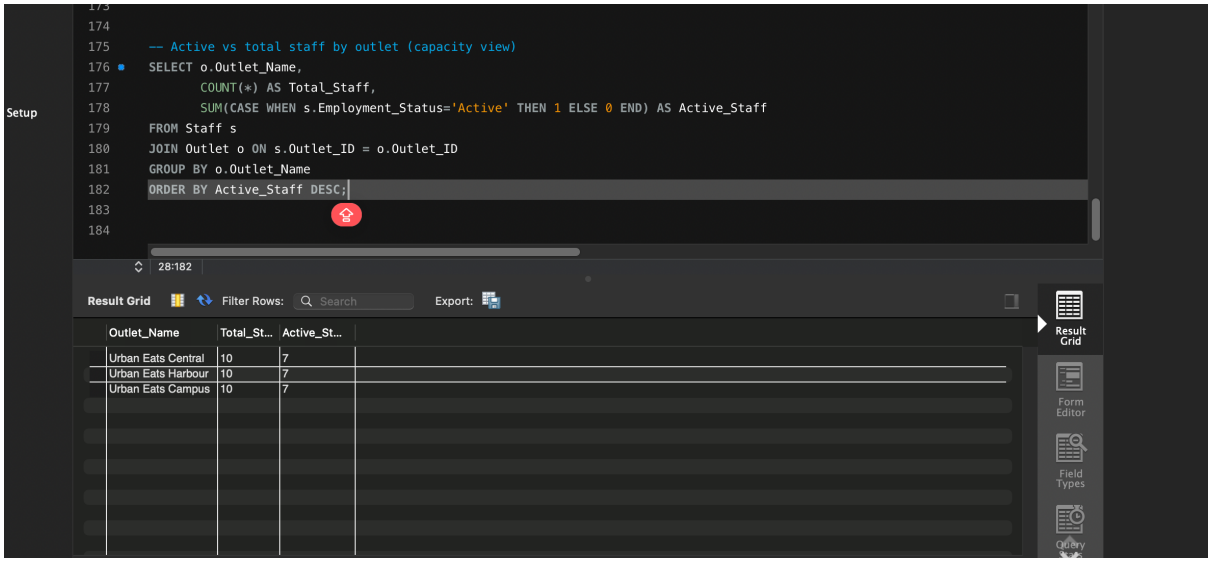
```
-- Shift utilisation: how full were shifts staffed vs Max_Staff
SELECT s.Shift_ID,
       s.Outlet_ID,
       s.Max_Staff,
       COUNT(ss.Staff_ID) AS Assigned,
       ROUND(100.0 * COUNT(ss.Staff_ID) / NULLIF(s.Max_Staff,0), 1) AS Utilisation_Pct
FROM Shift s
LEFT JOIN Staff_Shift ss ON ss.Shift_ID = s.Shift_ID
GROUP BY s.Shift_ID, s.Outlet_ID, s.Max_Staff
ORDER BY Utilisation_Pct ASC;
```

50% 30:163

Result Grid Filter Rows: Search Export:

Shift_ID	Outlet_ID	Max_Staff	Assigned	Utilisation_P...
3010	3	2	0	0.0
3009	3	2	0	0.0
2010	2	4	0	0.0
1009	1	6	0	0.0
1010	1	6	0	0.0
2009	2	5	0	0.0
1003	1	6	1	16.7
1006	1	6	1	16.7
1004	1	5	1	20.0
1005	1	5	1	20.0
1007	1	5	1	20.0
2007	2	5	1	20.0
2006	2	4	1	25.0
2005	2	4	1	25.0
2003	2	4	1	25.0
1008	1	4	1	25.0
2004	2	3	1	33.3
2008	2	3	1	33.3
3004	3	3	1	33.3
3006	3	3	1	33.3
3008	3	3	1	33.3
1002	1	6	2	33.3
2002	2	5	2	40.0
1001	1	5	2	40.0
2001	2	4	2	50.0
3003	3	2	1	50.0
3005	3	2	1	50.0
3007	3	2	1	50.0
3001	3	3	2	66.7
3002	3	3	2	66.7

Result 15 Read Only



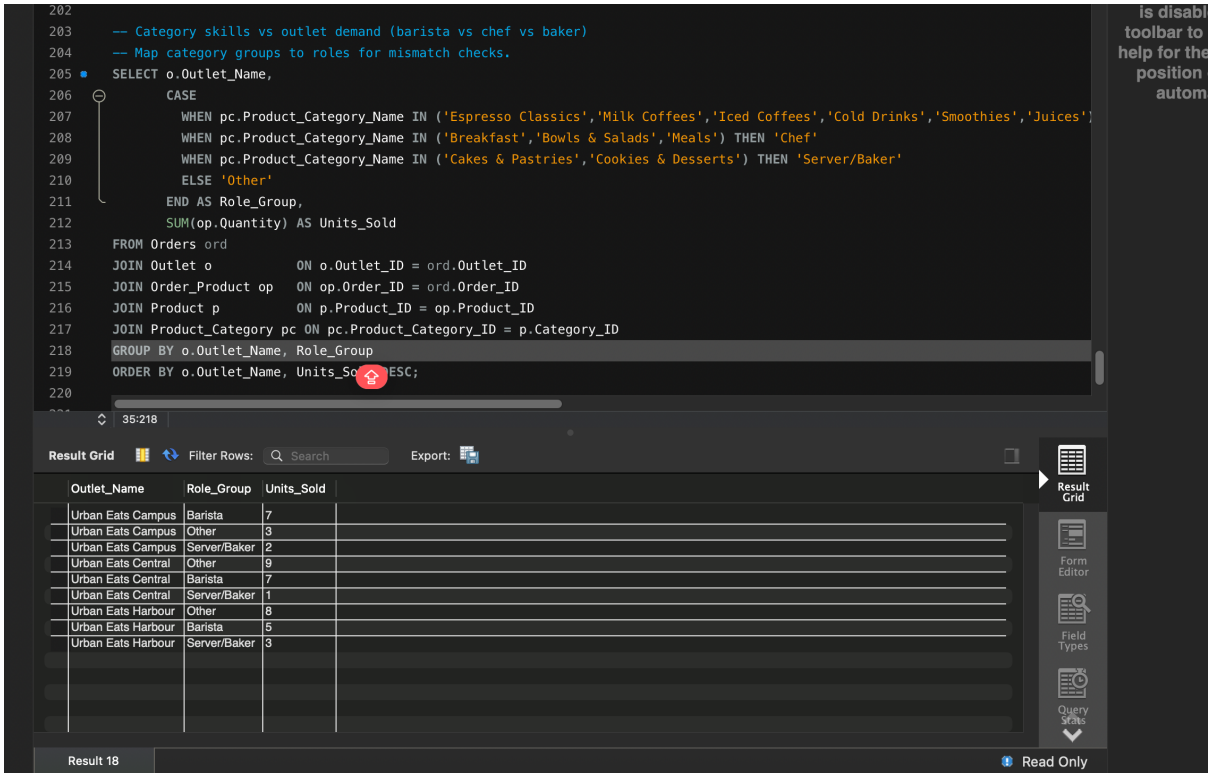
Setup

```
-- Active vs total staff by outlet (capacity view)
SELECT o.Outlet_Name,
       COUNT(*) AS Total_Staff,
       SUM(CASE WHEN s.Employment_Status='Active' THEN 1 ELSE 0 END) AS Active_Staff
FROM Staff s
JOIN Outlet o ON s.Outlet_ID = o.Outlet_ID
GROUP BY o.Outlet_Name
ORDER BY Active_Staff DESC;
```

28:182

Result Grid Filter Rows: Search Export:

Outlet_Name	Total_St...	Active_St...
Urban Eats Central	10	7
Urban Eats Harbour	10	7
Urban Eats Campus	10	7



Justification and Insights

Central shows strong, balanced role demand, but needs careful scheduling to ensure chef and barista coverage, with occasional underutilisation indicating potential overstaffing in certain shifts. Harbour demand is dominated by barista-driven categories such as cold drinks and coffees, aligning with its tourist-driven menu, and lower chef demand validates reduced chef staffing here. Campus faces severely under-utilised staff allocation with several shifts left unstaffed, leading to poor service reliability and an inability to capture revenue; Campus is also beverage-heavy, requiring primarily baristas and minimal chef input. Strategic actions include improving shift planning at Campus to eliminate 0% utilisation shifts and match staff presence with demand, optimising role allocation by outlet so that Central maintains balanced coverage across roles, Harbour focuses on baristas and servers while limiting chefs, and Campus scales down chefs while focusing on baristas, and exploring cross-training staff to cover gaps during under-staffed shifts to ensure resilience.

1.11 Business concern 5 Branch Level Operational Efficiency

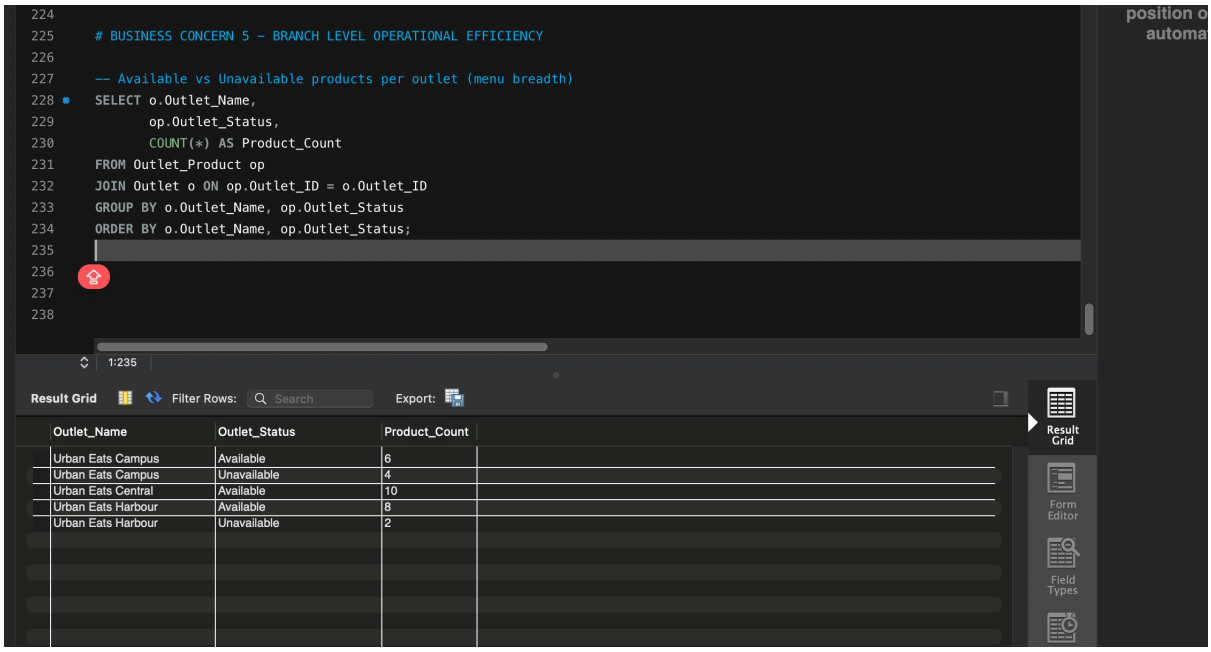
Name and Description: This concern evaluates how effectively each outlet manages product availability, reservation reliability, and the end-to-end order-to-payment flow. By comparing availability breadth, reservation conversion, and revenue outcomes, we can pinpoint where operational gaps are reducing outlet performance.

SQL Query

```
-- Available vs Unavailable products per outlet (menu breadth)
SELECT o.Outlet_Name,
    op.Outlet_Status,
    COUNT(*) AS Product_Count
FROM Outlet_Product op
JOIN Outlet o ON op.Outlet_ID = o.Outlet_ID
GROUP BY o.Outlet_Name, op.Outlet_Status
ORDER BY o.Outlet_Name, op.Outlet_Status;

-- Count of AVAILABLE products by outlet (quick capacity proxy)
SELECT o.Outlet_Name,
    COUNT(*) AS Available_Products
FROM Outlet_Product op
JOIN Outlet o ON op.Outlet_ID = o.Outlet_ID
WHERE op.Outlet_Status = 'Available'
GROUP BY o.Outlet_Name
ORDER BY Available_Products DESC;

-- Reservation conversion quality by outlet (service reliability)
SELECT o.Outlet_Name,
    r.Status,
    COUNT(*) AS Cnt
FROM Reservation r
JOIN Outlet o ON r.Outlet_ID = o.Outlet_ID
GROUP BY o.Outlet_Name, r.Status
ORDER BY o.Outlet_Name, Cnt DESC;

-- Orders and payments by outlet (end-to-end flow)
```

```
SELECT o.Outlet_Name,
    COUNT(DISTINCT ord.Order_ID)                    AS Total_Orders,
    SUM(CASE WHEN ord.Order_Status='Completed' THEN 1 ELSE 0 END) AS
Completed_Orders,
    SUM(p.Payment_Amount)                           AS Total_Revenue
FROM Orders ord
LEFT JOIN Payment p ON p.Order_ID = ord.Order_ID
JOIN Outlet o      ON o.Outlet_ID = ord.Outlet_ID
GROUP BY o.Outlet_Name
ORDER BY Total_Revenue DESC;
```

SQL Result



Classified

```sql
-- Count of AVAILABLE products by outlet (quick capacity proxy)
SELECT o.Outlet_Name,
       COUNT(*) AS Available_Products
FROM Outlet_Product op
JOIN Outlet o ON op.Outlet_ID = o.Outlet_ID
WHERE op.Outlet_Status = 'Available'
GROUP BY o.Outlet_Name
ORDER BY Available_Products DESC;
```

1:249

Result Grid Filter Rows: Search Export:

Outlet_Name	Available_Products
Urban Eats Central	10
Urban Eats Harbour	8
Urban Eats Campus	6

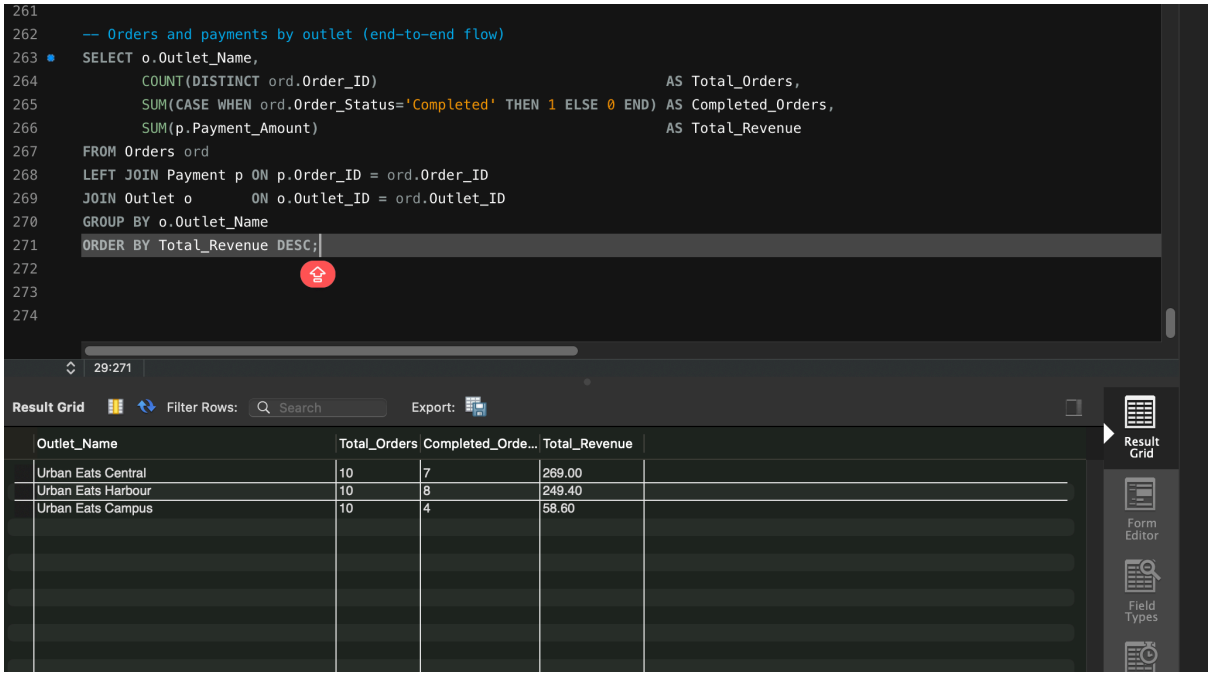
```sql
-- Reservation conversion quality by outlet (service reliability)
SELECT o.Outlet_Name,
       r.Status,
       COUNT(*) AS Cnt
FROM Reservation r
JOIN Outlet o ON r.Outlet_ID = o.Outlet_ID
GROUP BY o.Outlet_Name, r.Status
ORDER BY o.Outlet_Name, Cnt DESC;
```

1:259

Result Grid Filter Rows: Search Export:

Outlet_Name	Status	Cnt
Urban Eats Campus	Completed	4
Urban Eats Campus	No-show	3
Urban Eats Campus	Cancelled	2
Urban Eats Campus	Booked	1
Urban Eats Central	Completed	7
Urban Eats Central	Booked	2
Urban Eats Central	Cancelled	1
Urban Eats Harbour	Completed	8
Urban Eats Harbour	No-show	1
Urban Eats Harbour	Cancelled	1

Result 21



```
261
262   -- Orders and payments by outlet (end-to-end flow)
263   SELECT o.Outlet_Name,
264          COUNT(DISTINCT ord.Order_ID)                                  AS Total_Orders,
265          SUM(CASE WHEN ord.Order_Status='Completed' THEN 1 ELSE 0 END) AS Completed_Orders,
266          SUM(p.Payment_Amount)                                         AS Total_Revenue
267   FROM Orders ord
268   LEFT JOIN Payment p ON p.Order_ID = ord.Order_ID
269   JOIN Outlet o       ON o.Outlet_ID = ord.Outlet_ID
270   GROUP BY o.Outlet_Name
271   ORDER BY Total_Revenue DESC;
272
273
274
```

29:271

Result Grid | Filter Rows: Search | Export:

Outlet_Name	Total_Orders	Completed_Orde...	Total_Revenue
Urban Eats Central	10	7	269.00
Urban Eats Harbour	10	8	249.40
Urban Eats Campus	10	4	58.60

Result Grid | Form Editor | Field Types

Justification and Insights

Urban Eats Central is the strongest performer overall, with maximum menu breadth, high reservation reliability, and top revenue, showing that operational efficiency is aligned across availability and customer conversion. Urban Eats Harbour performs well despite slightly fewer products, benefiting from strong reservation conversion and demand for niche cold-drink and dessert items. Urban Eats Campus, however, is a major bottleneck with limited product availability, high cancellation and no-show rates, and the lowest revenue despite similar order inflow, which indicates that Campus suffers from poor execution rather than a shortage of demand. Strategic actions include expanding the available menu at Campus, enforcing stricter reservation and payment controls, and stabilising service reliability. Harbour should maintain its current strengths while improving stock consistency to match Central's product breadth, while Central should continue optimising as the operational benchmark and potentially serve as a best-practice model for Campus recovery.

1.12 Conclusion

Urban Eats demonstrates clear differences across its three outlets. Central consistently stands out as the most profitable and operationally balanced branch, offering a diverse menu, reliable reservations, and well-distributed staff roles, making it the benchmark for best practices. Harbour thrives in its niche of cold drinks and desserts, converting reservations effectively and leveraging its tourist-driven strategy, though its reliance on limited high-margin meals presents a growth challenge. Campus, in contrast, struggles with profitability due to high no-shows, cancellations, restricted product variety, and

poor staff allocation, highlighting execution issues rather than demand shortages. Across outlets, menu optimisation is necessary to address low-performing items, while customer retention remains fragile due to loyalty concentrated in single categories and payment failures linked to no-shows. Strategic focus should therefore centre on expanding meal offerings and process improvements at Campus, building loyalty incentives, refining staff scheduling, and pruning underperforming menu items. By reinforcing Central as the model of efficiency, Harbour as a niche-driven outlet, and Campus as the priority for operational recovery, Urban Eats can achieve stronger overall performance and sustainable growth.