BUSA8090 - Data and Visualisation for Business
Assignment 1

Urban Eats Café Chain: Analytics and Database Design Report

Student Name: NGOC BAO HAN TRAN

Student ID: 48393479

Part A – Database Design and Development

1.1 Entity Identification

Entity Name	Entity Description
PRODUCT_CATEGORY	A classification system for organizing menu items into logical groups such as beverages, meals, desserts, breakfast items, and snacks.
PRODUCT	Individual menu items offered by Urban Eats including food and beverage products. Each product has specific pricing, availability status, and belongs to a category.
CUSTOMER	Registered patrons who visit Urban Eats outlets, place orders, and may make reservations.
ORDERS	Transaction records representing customer purchases at Urban Eats outlets. Orders capture the complete transaction details including date, time, type (dine-in, takeaway, delivery), status, processed by a staff member, and total amount.
ORDER_ITEM	Associative entity linking orders to specific products with quantities and pricing details.
PAYMENT	Financial transaction records documenting how customers paid for their orders. Payment information includes amount, method (cash, card, digital), status, and refund tracking
RESERVATION	Booking records for customers who reserve tables at Urban Eats outlets.
OUTLET	Physical locations where Urban Eats operates, representing individual café branches. Each outlet has its own address, contact information, management structure, and operational data
STAFF	Employee records for all Urban Eats personnel including managers, chefs, servers, and baristas.
SHIFT	Work periods scheduled at specific outlets with defined start and end times.
STAFF_SHIFT	Associative entity tracking which staff members work specific shifts, including actual hours worked and attendance times.

1.2 Identify Attributes, Primary Keys and Foreign Keys

Entity Name	Attributes
PRODUCT_CATEGORY	**Primary Key:** Category_ID **Attributes:** Category_Name, Description
PRODUCT	**Primary Key:** Product_ID **Foreign Key:** • Category_ID → PRODUCT_CATEGORY (Category_ID) **Attributes:** Product_Name, Description, Price, Cost, Is_Available
CUSTOMER	**Primary Key:** Customer_ID **Attributes:** First_Name, Last_Name, Phone, Email, Address, Registration_Date, Last_Visit_Date
ORDERS	**Primary Key:** Order_ID **Foreign Keys:** • Customer_ID → CUSTOMER(Customer_ID) • Outlet_ID → OUTLET(Outlet_ID) **Attributes:** Order_Date, Order_Time, Order_Type, Status, Total_Amount
ORDER_ITEM	**Composite Primary Key/Foreign Keys:** • Order_ID → ORDERS(Order_ID) • Product_ID → PRODUCT(Product_ID) **Attributes:** Quantity, Unit_Price, Item_Cost
PAYMENT	**Primary Key:** Payment_ID **Foreign Key:** • Order_ID → ORDERS(Order_ID) **Attributes:** Payment_Amount, Payment_Method, Payment_Date, Payment_Status, Is_Refund
RESERVATION	**Primary Key:** Reservation_ID **Foreign Keys:** • Customer_ID → CUSTOMER(Customer_ID) • Outlet_ID → OUTLET(Outlet_ID) **Attributes:** Reservation_Date, Reservation_Time, Booked_Datetime, Party_Size, Status, Special_Requests
OUTLET	**Primary Key:** Outlet_ID **Foreign Key:** • Manager_ID → STAFF(Staff_ID) [Note: Nullable due to circular reference] **Attributes:** Outlet_Name, Phone, Address, State, Suburb, Opening_Date
STAFF	**Primary Key:** Staff_ID **Foreign Key:** • Home_Outlet_ID → OUTLET(Outlet_ID) **Attributes:** First_Name, Last_Name, Role, Hire_Date, Hourly_Rate
SHIFT	**Primary Key:** Shift_ID **Foreign Key:** Outlet_ID → OUTLET(Outlet_ID) **Attributes:** Shift_Date, Start_Time, End_Time, Shift_Type

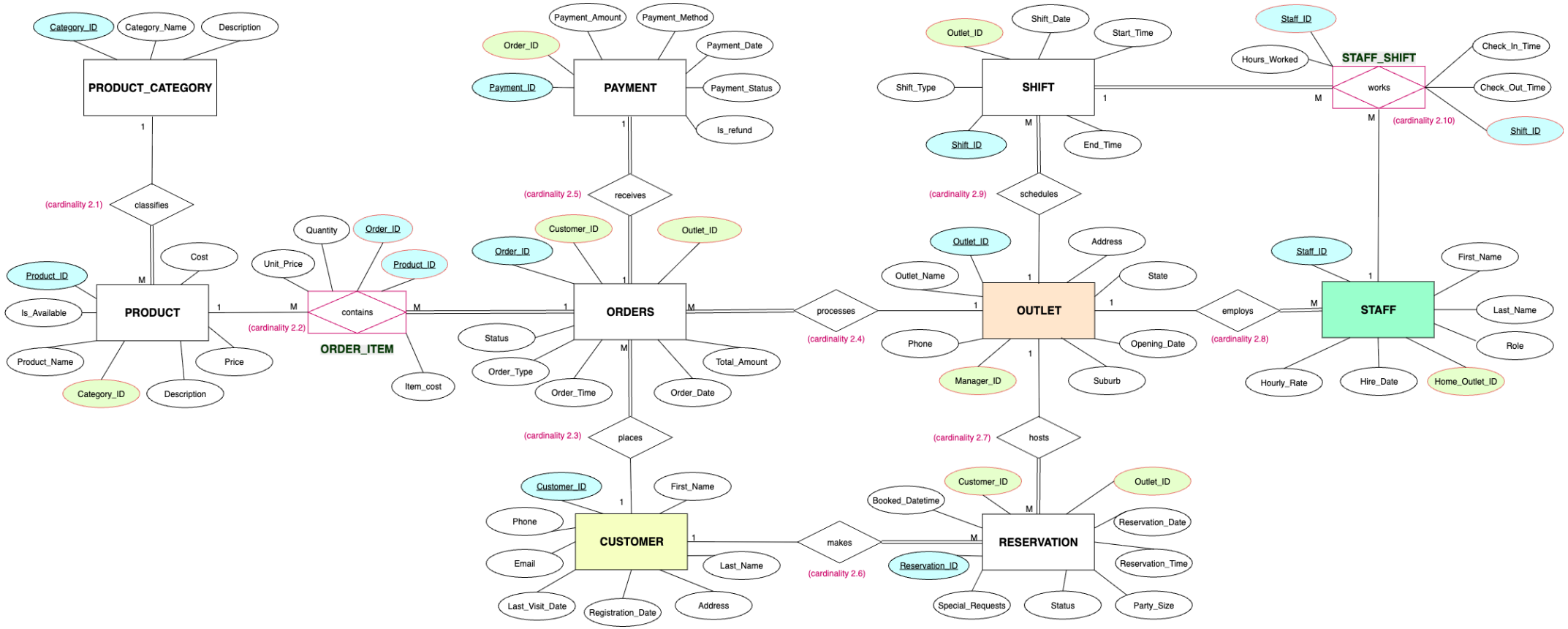
STAFF_SHIFT	**Composite Primary Key/Foreign Keys:** • Staff_ID → STAFF(Staff_ID) • Shift_ID → SHIFT(Shift_ID) **Attributes:** Check_In_Time, Check_Out_Time, Hours_Worked

1.3 Relationship Definition and Cardinality Analysis

Cardinality	Participation
2.1 PRODUCT_CATEGORY and PRODUCT: One-to-Many (1:M) • A PRODUCT_CATEGORY can classify zero or more PRODUCTS. • Every PRODUCT must belong to one PRODUCT_CATEGORY.	Partial on PRODUCT_CATEGORY side Total on PRODUCT side
2.2 PRODUCT and ORDERS: Many-to-Many through ORDER_ITEM (M:M) • A PRODUCT may be contained in zero or more ORDERS. • Every ORDER must contain at least one PRODUCT.	Partial on PRODUCT side Total on ORDERS side
2.3 CUSTOMER and ORDERS: One-to-Many (1:M) • A CUSTOMER may place zero or more ORDERS. • Every ORDER must be placed by one CUSTOMER.	Partial on CUSTOMER side Total on ORDERS side
2.4 OUTLET and ORDERS: One-to-Many (1:M) • An OUTLET may process zero or more ORDERS. • Every ORDER must be processed by one OUTLET.	Partial on OUTLET side Total on ORDERS side
2.5 ORDERS and PAYMENT: One-to-One (1:1) • Every completed ORDER must have a PAYMENT. • A PAYMENT cannot exist without an ORDER.	Total on ORDERS side Total on PAYMENT side
2.6 CUSTOMER and RESERVATION: One-to-Many (1:M) • A CUSTOMER may make zero or more RESERVATIONS. • Every RESERVATION must be made by one CUSTOMER.	Partial on CUSTOMER side Total on RESERVATION side
2.7 OUTLET and RESERVATION: One-to-Many (1:M) • An OUTLET may host zero or more RESERVATIONS • Every RESERVATION must be hosted by one OUTLET.	Partial on OUTLET side Total on RESERVATION side

2.8 OUTLET and STAFF: One-to-Many (1:M) • An OUTLET may employ zero or more STAFF. • Every STAFF member must be employed by one OUTLET.	Partial on OUTLET side Total on STAFF side
2.9 OUTLET and SHIFT: One-to-Many (1:M) • An OUTLET may schedule zero or more SHIFTS. • Every SHIFT must be scheduled by one OUTLET.	Partial on OUTLET side Total on SHIFT side
2.10 STAFF AND SHIFT: Many-to-Many through STAFF_SHIFT (M:M) • A STAFF member may work zero or more SHIFTS. • Every SHIFT must have at least one STAFF assigned.	Partial on STAFF side Total on SHIFT side

1.4 ERD diagram



1.5 Table implementation

Entities and Attributes	

PRODUCT_CATEGORY
• Category_ID (PK)
• Category_Name
• Description
	`-- 1. PRODUCT_CATEGORY Table`
	`CREATE TABLE PRODUCT_CATEGORY (`
	`  Category_ID INT PRIMARY KEY AUTO_INCREMENT,`
	`  Category_Name VARCHAR(50) NOT NULL UNIQUE,`
	`  Description TEXT`
	`);`

PRODUCT
• Product_ID (PK)
• Category_ID (FK → PRODUCT_CATEGORY.Category_ID)
• Product_Name
• Description
• Price
• Cost
• Is_Available
	`-- 2. PRODUCT Table`
	`CREATE TABLE PRODUCT (`
	`  Product_ID INT PRIMARY KEY AUTO_INCREMENT,`
	`  Category_ID INT NOT NULL,`
	`  Product_Name VARCHAR(100) NOT NULL,`
	`  Description TEXT,`
	`  Price DECIMAL(8 , 2 ) NOT NULL CHECK (Price >= 0),`
	`  Cost DECIMAL(8 , 2 ) NOT NULL CHECK (Cost >= 0),`
	`  Is_Available BOOLEAN DEFAULT TRUE,`
	`  FOREIGN KEY (Category_ID)`
	`  REFERENCES PRODUCT_CATEGORY (Category_ID),`
	`  CONSTRAINT chk_profit CHECK (Price >= Cost)`
	`);`

CUSTOMER
• Customer_ID (PK)
• First_Name
• Last_Name
• Phone
• Email
• Address
• Registration_Date
• Last_Visit_Date
	`-- 3. CUSTOMER Table`
	`CREATE TABLE CUSTOMER (`
	`  Customer_ID INT PRIMARY KEY AUTO_INCREMENT,`
	`  First_Name VARCHAR(50) NOT NULL,`
	`  Last_Name VARCHAR(50) NOT NULL,`
	`  Phone VARCHAR(15) UNIQUE,`
	`  Email VARCHAR(100) UNIQUE,`
	`  Address TEXT,`
	`  Registration_Date DATE NOT NULL DEFAULT (CURRENT_DATE),`
	`  Last_Visit_Date DATE,`
	`  CONSTRAINT chk_visit_after_reg`
	`  CHECK (Last_Visit_Date >= Registration_Date OR Last_Visit_Date IS NULL)`
	`);`

OUTLET
• Outlet_ID (PK)
• Manager_ID (FK → STAFF.Staff_ID, Nullable due to circular reference)
• Outlet_Name
• Phone
• Address
• State
• Suburb
• Opening_Date
	`-- 4. OUTLET Table (Created before STAFF to handle circular reference)`
	`CREATE TABLE OUTLET (`
	`  Outlet_ID INT PRIMARY KEY AUTO_INCREMENT,`
	`  Outlet_Name VARCHAR(100) NOT NULL,`
	`  Phone VARCHAR(15) NOT NULL,`
	`  Address TEXT NOT NULL,`
	`  State VARCHAR(50) NOT NULL,`
	`  Suburb VARCHAR(50) NOT NULL,`
	`  Opening_Date DATE NOT NULL,`
	`  Manager_ID INT NULL`
	`);`

STAFF
- Staff_ID (PK)
- Home_Outlet_ID (FK → OUTLET.Outlet_ID)
- First_Name
- Last_Name
- Role
- Hire_Date
- Hourly_Rate

```sql
-- 5. STAFF Table
CREATE TABLE STAFF (
    Staff_ID INT PRIMARY KEY AUTO_INCREMENT,
    First_Name VARCHAR(50) NOT NULL,
    Last_Name VARCHAR(50) NOT NULL,
    Role ENUM('Manager', 'Chef', 'Barista', 'Server', 'Kitchen Staff')
    NOT NULL,
    Hire_Date DATE NOT NULL DEFAULT (CURRENT_DATE),
    Hourly_Rate DECIMAL(5,2) NOT NULL CHECK (Hourly_Rate >= 0),
    Home_Outlet_ID INT NOT NULL,
    FOREIGN KEY (Home_Outlet_ID) REFERENCES OUTLET(Outlet_ID)
);

-- Add Manager_ID foreign key to OUTLET after STAFF creation
ALTER TABLE OUTLET
ADD CONSTRAINT fk_outlet_manager
FOREIGN KEY (Manager_ID) REFERENCES STAFF(Staff_ID);
```

ORDERS
- Order_ID (PK)
- Customer_ID (FK → CUSTOMER.Customer_ID)
- Outlet_ID (FK → OUTLET.Outlet_ID)
- Order_Date
- Order_Time
- Order_Type
- Status
- Total_Amount

```sql
-- 6. ORDERS Table
CREATE TABLE ORDERS (
    Order_ID INT PRIMARY KEY AUTO_INCREMENT,
    Customer_ID INT NOT NULL,
    Outlet_ID INT NOT NULL,
    Order_Date DATE NOT NULL DEFAULT (CURRENT_DATE),
    Order_Time TIME NOT NULL DEFAULT (CURRENT_TIME),
    Order_Type ENUM('Dine-in', 'Takeaway', 'Delivery') NOT NULL,
    Status ENUM('Pending', 'Preparing', 'Ready', 'Completed', 'Cancelled')
    DEFAULT 'Pending',
    Total_Amount DECIMAL(10,2) NOT NULL CHECK (Total_Amount >= 0),
    FOREIGN KEY (Customer_ID) REFERENCES CUSTOMER(Customer_ID),
    FOREIGN KEY (Outlet_ID) REFERENCES OUTLET(Outlet_ID)
);
```

ORDER_ITEM
- Order_ID (PK, FK → ORDERS.Order_ID)
- Product_ID (PK, FK → PRODUCT.Product_ID)
- Quantity
- Unit_Price
- Item_Cost

```sql
-- 7. ORDER_ITEM Table (Associative Entity)
CREATE TABLE ORDER_ITEM (
    Order_ID INT NOT NULL,
    Product_ID INT NOT NULL,
    Quantity INT NOT NULL CHECK (Quantity > 0),
    Unit_Price DECIMAL(8 , 2 ) NOT NULL CHECK (Unit_Price >= 0),
    Item_Cost DECIMAL(8 , 2 ) NOT NULL CHECK (Item_Cost >= 0),
    PRIMARY KEY (Order_ID , Product_ID),
    FOREIGN KEY (Order_ID)
        REFERENCES ORDERS (Order_ID),
    FOREIGN KEY (Product_ID)
        REFERENCES PRODUCT (Product_ID)
);
```

PAYMENT
- Payment_ID (PK)
- Order_ID (FK → ORDERS.Order_ID)
- Payment_Amount
- Payment_Method
- Payment_Date
- Payment_Status

Is_Refund

```sql
-- 8. PAYMENT Table
CREATE TABLE PAYMENT (
    Payment_ID INT PRIMARY KEY AUTO_INCREMENT,
    Order_ID INT NOT NULL UNIQUE,
    Payment_Amount DECIMAL(10,2) NOT NULL CHECK (Payment_Amount >= 0),
    Payment_Method ENUM('Cash', 'Credit Card', 'Debit Card', 'Digital Wallet', 'EFTPOS')
    NOT NULL,
    Payment_Date DATE NOT NULL DEFAULT (CURRENT_DATE),
    Payment_Status ENUM('Pending', 'Completed', 'Failed', 'Refunded')
    NOT NULL DEFAULT 'Pending',
    Is_Refund BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (Order_ID) REFERENCES ORDERS (Order_ID)
);
```

RESERVATION
- Reservation_ID (PK)
- Customer_ID (FK → CUSTOMER.Customer_ID)
- Outlet_ID (FK → OUTLET.Outlet_ID)
- Reservation_Date
- Reservation_Time
- Booked_Datetime
- Party_Size
- Status
- Special_Requests

```sql
-- 9. RESERVATION Table
CREATE TABLE RESERVATION (
    Reservation_ID INT PRIMARY KEY AUTO_INCREMENT,
    Customer_ID INT NOT NULL,
    Outlet_ID INT NOT NULL,
    Reservation_Date DATE NOT NULL,
    Reservation_Time TIME NOT NULL,
    Booked_Datetime TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Party_Size INT NOT NULL CHECK (Party_Size > 0),
    Status ENUM('Confirmed', 'Cancelled', 'Completed', 'No-show')
    DEFAULT 'Confirmed',
    Special_Requests TEXT,
    FOREIGN KEY (Customer_ID) REFERENCES CUSTOMER(Customer_ID),
    FOREIGN KEY (Outlet_ID) REFERENCES OUTLET(Outlet_ID)
);
```

SHIFT
- Shift_ID (PK)
- Outlet_ID (FK → OUTLET.Outlet_ID)
- Shift_Date
- Start_Time
- End_Time
- Shift_Type

```sql
-- 10. SHIFT Table
CREATE TABLE SHIFT (
    Shift_ID INT PRIMARY KEY AUTO_INCREMENT,
    Outlet_ID INT NOT NULL,
    Shift_Date DATE NOT NULL,
    Start_Time TIME NOT NULL,
    End_Time TIME NOT NULL,
    Shift_Type ENUM('Morning', 'Afternoon', 'Evening', 'Night') NOT NULL,
    FOREIGN KEY (Outlet_ID)
        REFERENCES OUTLET (Outlet_ID),
    CONSTRAINT chk_shift_times CHECK (End_Time > Start_Time)
);
```

STAFF_SHIFT
- Staff_ID (PK, FK → STAFF.Staff_ID)
- Shift_ID (PK, FK → SHIFT.Shift_ID)
- Check_In_Time
- Check_Out_Time
- Hours_Worked

```sql
-- 11. STAFF_SHIFT Table (Associative Entity)
CREATE TABLE STAFF_SHIFT (
    Staff_ID INT NOT NULL,
    Shift_ID INT NOT NULL,
    Check_In_Time DATETIME,
    Check_Out_Time DATETIME,
    Hours_Worked DECIMAL(4 , 2 ) DEFAULT 0,
    PRIMARY KEY (Staff_ID , Shift_ID),
    FOREIGN KEY (Staff_ID)
        REFERENCES STAFF (Staff_ID),
    FOREIGN KEY (Shift_ID)
        REFERENCES SHIFT (Shift_ID),
    CONSTRAINT chk_checkout_after_checkin
        CHECK (Check_Out_Time > Check_In_Time OR Check_Out_Time IS NULL)
);
```

1.6 Dummy Data insertion

- PRODUCT_CATEGORY table

```
-- 1. PRODUCT_CATEGORY Table (15 rows)
INSERT INTO PRODUCT_CATEGORY (Category_Name, Description) VALUES
('Hot Beverages', 'Coffee, tea, hot chocolate and other warm drinks'),
('Cold Beverages', 'Iced drinks, smoothies, juices and cold refreshments'),
('Breakfast Items', 'Morning meals including eggs, toast, pastries and cereals'),
('Lunch Mains', 'Hearty lunch dishes including burgers, salads, and sandwiches'),
('Light Meals', 'Smaller portions and snacks for quick dining'),
('Desserts', 'Sweet treats including cakes, cookies, and ice cream'),
('Pastries', 'Fresh baked goods including croissants, muffins, and danishes'),
('Soups', 'Hot and cold soup varieties'),
('Salads', 'Fresh garden salads and healthy options'),
('Sides', 'Complementary items like fries, bread, and appetizers'),
('Healthy Options', 'Low-calorie and nutritious menu items'),
('Kids Menu', 'Child-friendly meals and portions'),
('Seasonal Specials', 'Limited-time seasonal menu items'),
('Vegan Options', 'Plant-based menu items'),
('Gluten-Free', 'Items suitable for gluten-free diets');
```

- PRODUCT table

```
-- 2. PRODUCT Table ( key entity)
INSERT INTO PRODUCT (Category_ID, Product_Name, Description, Price, Cost, Is_Available) VALUES
-- Hot Beverages
(1, 'Cappuccino', 'Rich espresso with steamed milk foam', 4.50, 1.20, TRUE),
(1, 'Latte', 'Smooth espresso with steamed milk', 4.80, 1.30, TRUE),
(1, 'Flat White', 'Double shot espresso with microfoam milk', 4.20, 1.15, TRUE),
(1, 'Hot Chocolate', 'Premium chocolate with whipped cream', 5.20, 1.80, TRUE),
(1, 'Earl Grey Tea', 'Traditional bergamot-infused black tea', 3.80, 0.90, TRUE),
-- Cold Beverages
(2, 'Iced Coffee', 'Cold brew coffee served over ice', 4.20, 1.25, TRUE),
(2, 'Green Smoothie', 'Spinach, apple, banana and honey blend', 6.50, 2.10, TRUE),
(2, 'Fresh Orange Juice', 'Freshly squeezed orange juice', 4.80, 1.60, TRUE),
(2, 'Iced Tea', 'Refreshing iced tea with lemon', 3.50, 0.95, TRUE),
-- Breakfast Items
(3, 'Big Breakfast', 'Eggs, bacon, sausage, beans, toast, hashbrown', 18.90, 7.50, TRUE),
(3, 'Avocado Toast', 'Smashed avocado on sourdough with feta', 12.50, 4.20, TRUE),
(3, 'Pancakes Stack', 'Three fluffy pancakes with maple syrup', 14.20, 4.80, TRUE),
(3, 'Breakfast Burrito', 'Scrambled eggs, cheese, bacon in tortilla', 11.80, 4.10, TRUE),
-- Lunch Mains
(4, 'Gourmet Burger', 'Beef patty with premium toppings and fries', 19.50, 7.80, TRUE),
(4, 'Caesar Salad', 'Cos lettuce, parmesan, croutons, caesar dressing', 15.90, 5.30, TRUE),
(4, 'Fish and Chips', 'Beer battered fish with chunky chips', 22.90, 9.20, TRUE),
(4, 'Chicken Wrap', 'Grilled chicken with salad in tortilla wrap', 13.80, 4.90, TRUE),
-- Light Meals
(5, 'Soup of the Day', 'Daily rotating soup with bread roll', 8.90, 2.80, TRUE),
(5, 'Bruschetta', 'Toasted bread with tomato, basil, balsamic', 9.50, 2.90, TRUE),
(5, 'Cheese Toastie', 'Grilled cheese sandwich with tomato', 7.80, 2.20, TRUE),
-- Desserts
(6, 'Chocolate Brownie', 'Rich chocolate brownie with vanilla ice cream', 8.90, 2.60, TRUE),
(6, 'Cheesecake Slice', 'New York style cheesecake with berry coulis', 9.50, 3.10, TRUE),
(6, 'Apple Pie', 'Traditional apple pie with custard', 8.20, 2.40, TRUE),
-- Pastries
(7, 'Croissant', 'Buttery French croissant', 4.20, 1.30, TRUE),
(7, 'Blueberry Muffin', 'Fresh blueberry muffin', 4.80, 1.50, TRUE),
(7, 'Danish Pastry', 'Flaky pastry with fruit filling', 5.20, 1.70, TRUE),
-- Healthy Options
(11, 'Acai Bowl', 'Acai berries with granola and fresh fruit', 12.90, 4.50, TRUE),
(11, 'Quinoa Salad', 'Quinoa with roasted vegetables and dressing', 14.50, 5.20, TRUE),
-- Vegan Options
(14, 'Vegan Burger', 'Plant-based patty with vegan cheese and fries', 17.90, 6.80, TRUE),
(14, 'Coconut Curry', 'Vegetables in coconut curry sauce with rice', 16.50, 5.90, TRUE),
-- Seasonal (currently unavailable)
(13, 'Pumpkin Spice Latte', 'Autumn spiced latte with pumpkin flavor', 5.50, 1.90, FALSE);
```

Classified

```sql
-- 3. CUSTOMER Table (25 rows)
INSERT INTO CUSTOMER (First_Name, Last_Name, Phone, Email, Address, Registration_Date, Last_Visit_Date) VALUES
('Sarah', 'Johnson', '0412345678', 'sarah.johnson@email.com', '123 Main St, Sydney NSW 2000', '2024-01-15', '2024-09-15'),
('Michael', 'Chen', '0423456789', 'michael.chen@email.com', '456 George St, Sydney NSW 2000', '2024-02-20', '2024-09-10'),
('Emma', 'Williams', '0434567890', 'emma.williams@email.com', '789 Pitt St, Sydney NSW 2000', '2024-01-10', '2024-09-12'),
('James', 'Brown', '0445678901', 'james.brown@email.com', '321 Collins St, Melbourne VIC 3000', '2024-03-05', '2024-09-08'),
('Lisa', 'Davis', '0456789012', 'lisa.davis@email.com', '654 Flinders St, Melbourne VIC 3000', '2024-02-28', '2024-09-14'),
('David', 'Wilson', '0467890123', 'david.wilson@email.com', '987 Queen St, Brisbane QLD 4000', '2024-01-25', '2024-09-11'),
('Sophie', 'Taylor', '0478901234', 'sophie.taylor@email.com', '147 King St, Sydney NSW 2000', '2024-04-12', '2024-09-13'),
('Ryan', 'Anderson', '0489012345', 'ryan.anderson@email.com', '258 Bourke St, Melbourne VIC 3000', '2024-03-18', '2024-08-20'),
('Jessica', 'Thomas', '0490123456', 'jessica.thomas@email.com', '369 Ann St, Brisbane QLD 4000', '2024-02-14', '2024-09-09'),
('Matthew', 'Jackson', '0401234567', 'matthew.jackson@email.com', '741 Elizabeth St, Sydney NSW 2000', '2024-05-03', '2024-09-16'),
('Amy', 'White', '0412345679', 'amy.white@email.com', '852 Little Collins St, Melbourne VIC 3000', '2024-01-30', '2024-07-15'),
('Daniel', 'Harris', '0423456780', 'daniel.harris@email.com', '963 Edward St, Brisbane QLD 4000', '2024-04-08', '2024-09-07'),
('Rachel', 'Martin', '0434567891', 'rachel.martin@email.com', '159 York St, Sydney NSW 2000', '2024-03-22', '2024-09-05'),
('Andrew', 'Thompson', '0445678902', 'andrew.thompson@email.com', '357 Swanston St, Melbourne VIC 3000', '2024-02-10', '2024-06-30'),
('Chloe', 'Garcia', '0456789013', 'chloe.garcia@email.com', '468 Adelaide St, Brisbane QLD 4000', '2024-05-15', '2024-09-03'),
('Benjamin', 'Martinez', '0467890124', 'ben.martinez@email.com', '579 Kent St, Sydney NSW 2000', '2024-01-08', '2024-09-01'),
('Olivia', 'Robinson', '0478901235', 'olivia.robinson@email.com', '680 La Trobe St, Melbourne VIC 3000', '2024-04-25', '2024-08-28'),
('Nathan', 'Clark', '0489012346', 'nathan.clark@email.com', '791 Queen St, Brisbane QLD 4000', '2024-03-12', '2024-08-15'),
('Isabella', 'Rodriguez', '0490123457', 'isabella.rodriguez@email.com', '802 Castlereagh St, Sydney NSW 2000', '2024-05-20', '2024-08-10'),
('Joshua', 'Lewis', '0401234568', 'joshua.lewis@email.com', '913 Russell St, Melbourne VIC 3000', '2024-02-05', '2024-07-25'),
('Grace', 'Lee', '0412345680', 'grace.lee@email.com', '124 Creek St, Brisbane QLD 4000', '2024-06-01', '2024-09-17'),
('Ethan', 'Walker', '0423456781', 'ethan.walker@email.com', '235 Park St, Sydney NSW 2000', '2024-04-30', '2024-09-18'),
('Mia', 'Hall', '0434567892', 'mia.hall@email.com', '346 Spencer St, Melbourne VIC 3000', '2024-05-10', '2024-09-16'),
('Lucas', 'Allen', '0445678903', 'lucas.allen@email.com', '457 Charlotte St, Brisbane QLD 4000', '2024-03-28', '2024-08-05'),
('Zoe', 'Young', '0456789014', 'zoe.young@email.com', '568 Macquarie St, Sydney NSW 2000', '2024-06-15', '2024-09-14');
```

- OUTLET table

```sql
-- 4. OUTLET Table (3 outlets as per business requirement)
INSERT INTO OUTLET (Outlet_Name, Phone, Address, State, Suburb, Opening_Date, Manager_ID) VALUES
('Urban Eats CBD Sydney', '0287654321', '100 George Street, Sydney NSW 2000', 'NSW', 'Sydney CBD', '2023-03-15', NULL),
('Urban Eats South Melbourne', '0398765432', '200 Collins Street, Melbourne VIC 3000', 'VIC', 'South Melbourne', '2023-06-10', NULL),
('Urban Eats Brisbane City', '0732109876', '300 Queen Street, Brisbane QLD 4000', 'QLD', 'Brisbane City', '2023-09-20', NULL);
```

- STAFF table

```sql
-- 5. STAFF Table
INSERT INTO STAFF (First_Name, Last_Name, Role, Hire_Date, Hourly_Rate, Home_Outlet_ID) VALUES
-- Sydney Staff
('Tom', 'Mitchell', 'Manager', '2023-03-10', 35.00, 1),
('Maria', 'Santos', 'Chef', '2023-03-20', 28.50, 1),
('Jake', 'Thompson', 'Barista', '2023-04-01', 24.00, 1),
('Emma', 'Kelly', 'Server', '2023-04-15', 22.50, 1),
('Alex', 'Nguyen', 'Kitchen Staff', '2023-05-01', 23.00, 1),
('Sophie', 'Green', 'Barista', '2023-05-15', 24.00, 1),
('Ryan', 'O\'Connor', 'Server', '2023-06-01', 22.50, 1),
-- Melbourne Staff
('Lisa', 'Parker', 'Manager', '2023-06-05', 35.00, 2),
('Marco', 'Rossi', 'Chef', '2023-06-15', 29.00, 2),
('Chloe', 'Wong', 'Barista', '2023-07-01', 24.50, 2),
('James', 'Stewart', 'Server', '2023-07-15', 22.50, 2),
('Priya', 'Patel', 'Kitchen Staff', '2023-08-01', 23.50, 2),
('Daniel', 'Kim', 'Barista', '2023-08-15', 24.50, 2),
-- Brisbane Staff
('Sarah', 'Clarke', 'Manager', '2023-09-15', 34.50, 3),
('Tony', 'Moretti', 'Chef', '2023-09-25', 28.00, 3),
('Jessica', 'Liu', 'Barista', '2023-10-01', 24.00, 3),
('Michael', 'Fraser', 'Server', '2023-10-15', 22.50, 3),
('Amy', 'Davis', 'Kitchen Staff', '2023-11-01', 23.00, 3),
('Chris', 'Taylor', 'Server', '2023-11-15', 22.50, 3),
('Zoe', 'Campbell', 'Barista', '2024-01-10', 24.00, 3);
-- Update Manager_ID in OUTLET table
UPDATE OUTLET SET Manager_ID = 1 WHERE Outlet_ID = 1;
UPDATE OUTLET SET Manager_ID = 8 WHERE Outlet_ID = 2;
UPDATE OUTLET SET Manager_ID = 14 WHERE Outlet_ID = 3;
```

- ORDERS table

```sql
-- 6. ORDERS Table (key entity)
INSERT INTO ORDERS (Customer_ID, Outlet_ID, Order_Date, Order_Time, Order_Type, Status, Total_Amount) VALUES
-- Recent orders (varied times and types)
(1, 1, '2024-09-18', '08:30:00', 'Dine-in', 'Completed', 23.40),
(2, 1, '2024-09-18', '09:15:00', 'Takeaway', 'Completed', 9.30),
(3, 1, '2024-09-18', '12:45:00', 'Delivery', 'Completed', 31.80),
(4, 2, '2024-09-18', '07:45:00', 'Dine-in', 'Completed', 18.90),
(5, 2, '2024-09-18', '13:20:00', 'Takeaway', 'Completed', 15.60),
(6, 3, '2024-09-17', '08:15:00', 'Dine-in', 'Completed', 26.70),
(7, 1, '2024-09-17', '14:30:00', 'Delivery', 'Completed', 19.50),
(8, 2, '2024-09-17', '11:00:00', 'Takeaway', 'Completed', 12.80),
(9, 3, '2024-09-16', '16:45:00', 'Dine-in', 'Completed', 17.40),
(10, 1, '2024-09-16', '10:30:00', 'Takeaway', 'Completed', 14.20),
(11, 2, '2024-09-15', '09:00:00', 'Dine-in', 'Completed', 22.90),
(12, 3, '2024-09-15', '13:15:00', 'Delivery', 'Completed', 28.40),
(13, 1, '2024-09-14', '15:30:00', 'Takeaway', 'Completed', 8.90),
(14, 2, '2024-09-14', '12:00:00', 'Dine-in', 'Completed', 33.50),
(15, 3, '2024-09-13', '08:45:00', 'Takeaway', 'Completed', 16.70),
-- Older orders for analytics
(16, 1, '2024-09-10', '11:30:00', 'Dine-in', 'Completed', 25.30),
(17, 2, '2024-09-08', '14:15:00', 'Delivery', 'Completed', 21.60),
(18, 3, '2024-09-05', '09:30:00', 'Takeaway', 'Completed', 13.40),
(19, 1, '2024-08-28', '16:00:00', 'Dine-in', 'Completed', 29.80),
(20, 2, '2024-08-25', '10:45:00', 'Takeaway', 'Completed', 18.20),
-- Some recent orders in different statuses
(21, 1, '2024-09-18', '18:30:00', 'Dine-in', 'Preparing', 24.70),
(22, 2, '2024-09-18', '19:00:00', 'Delivery', 'Ready', 32.40),
(23, 3, '2024-09-18', '17:45:00', 'Takeaway', 'Pending', 11.80),
(1, 2, '2024-09-12', '08:00:00', 'Dine-in', 'Completed', 21.40),
(3, 3, '2024-09-10', '12:30:00', 'Takeaway', 'Completed', 15.90),
(5, 1, '2024-09-08', '14:45:00', 'Delivery', 'Completed', 27.30),
(7, 2, '2024-09-06', '11:15:00', 'Dine-in', 'Completed', 19.60),
(9, 3, '2024-09-04', '15:00:00', 'Takeaway', 'Completed', 23.80),
(11, 1, '2024-09-02', '09:45:00', 'Dine-in', 'Completed', 16.50),
(13, 2, '2024-08-30', '13:30:00', 'Delivery', 'Completed', 22.10);
```

- ORDER_ITEM table

```sql
-- 7. ORDER_ITEM Table ( associative entity with multiple items per order)
INSERT INTO ORDER_ITEM (Order_ID, Product_ID, Quantity, Unit_Price, Item_Cost) VALUES
-- Order 1 items (Customer 1, Total: 23.40)
(1, 2, 1, 4.80, 1.30),  -- Latte
(1, 10, 1, 18.90, 7.50), -- Big Breakfast
-- Order 2 items (Customer 2, Total: 9.30)
(2, 1, 2, 4.50, 1.20),   -- 2 Cappuccinos
-- Order 3 items (Customer 3, Total: 31.80)
(3, 14, 1, 19.50, 7.80), -- Gourmet Burger
(3, 21, 1, 8.90, 2.60),  -- Chocolate Brownie
(3, 6, 1, 4.20, 1.25),   -- Iced Coffee
-- Order 4 items (Customer 4, Total: 18.90)
(4, 10, 1, 18.90, 7.50), -- Big Breakfast
-- Order 5 items (Customer 5, Total: 15.60)
(5, 15, 1, 15.90, 5.30), -- Caesar Salad
-- Order 6 items (Customer 6, Total: 26.70)
(6, 12, 1, 14.20, 4.80), -- Pancakes
(6, 1, 1, 4.50, 1.20),   -- Cappuccino
(6, 7, 1, 6.50, 2.10),   -- Green Smoothie
(6, 23, 1, 4.20, 1.30),  -- Croissant
-- Order 7 items (Customer 7, Total: 19.50)
(7, 14, 1, 19.50, 7.80), -- Gourmet Burger
-- Order 8 items (Customer 8, Total: 12.80)
(8, 11, 1, 12.50, 4.20), -- Avocado Toast
-- Order 9 items (Customer 9, Total: 17.40)
(9, 17, 1, 13.80, 4.90), -- Chicken Wrap
(9, 9, 1, 3.50, 0.95),   -- Iced Tea
-- Order 10 items (Customer 10, Total: 14.20)
(10, 12, 1, 14.20, 4.80), -- Pancakes
```

Classified

```
-- Continue with more order items...
(11, 16, 1, 22.90, 9.20), -- Fish and Chips
(12, 27, 1, 12.90, 4.50), -- Acai Bowl
(12, 2, 1, 4.80, 1.30),   -- Latte
(12, 24, 1, 4.80, 1.50),  -- Blueberry Muffin
(13, 18, 1, 8.90, 2.80),  -- Soup of the Day
(14, 28, 1, 14.50, 5.20), -- Quinoa Salad
(14, 29, 1, 17.90, 6.80), -- Vegan Burger
(14, 6, 1, 4.20, 1.25),   -- Iced Coffee
(15, 13, 1, 11.80, 4.10), -- Breakfast Burrito
(15, 3, 1, 4.20, 1.15),   -- Flat White
(16, 30, 1, 16.50, 5.90), -- Coconut Curry
(16, 8, 1, 4.80, 1.60),   -- Orange Juice
(17, 19, 1, 9.50, 2.90),  -- Bruschetta
(17, 22, 1, 9.50, 3.10),  -- Cheesecake
(17, 1, 1, 4.50, 1.20),   -- Cappuccino
(18, 20, 1, 7.80, 2.20),  -- Cheese Toastie
(18, 4, 1, 5.20, 1.80),   -- Hot Chocolate
(19, 21, 2, 8.90, 2.60),  -- 2 Chocolate Brownies
(19, 26, 1, 5.20, 1.70),  -- Danish Pastry
(19, 2, 1, 4.80, 1.30),   -- Latte
(20, 11, 1, 12.50, 4.20), -- Avocado Toast
(20, 5, 1, 3.80, 0.90),   -- Earl Grey Tea
-- Current orders (non-completed)
(21, 15, 1, 15.90, 5.30), -- Caesar Salad
(21, 1, 2, 4.50, 1.20),   -- 2 Cappuccinos
(22, 14, 1, 19.50, 7.80), -- Gourmet Burger
(22, 21, 1, 8.90, 2.60),  -- Chocolate Brownie
(22, 6, 1, 4.20, 1.25),   -- Iced Coffee
(23, 13, 1, 11.80, 4.10), -- Breakfast Burrito
```

```
-- Current orders (non-completed)
(21, 15, 1, 15.90, 5.30), -- Caesar Salad
(21, 1, 2, 4.50, 1.20),   -- 2 Cappuccinos
(22, 14, 1, 19.50, 7.80), -- Gourmet Burger
(22, 21, 1, 8.90, 2.60),  -- Chocolate Brownie
(22, 6, 1, 4.20, 1.25),   -- Iced Coffee
(23, 13, 1, 11.80, 4.10), -- Breakfast Burrito
-- Additional completed orders
(24, 17, 1, 13.80, 4.90), -- Chicken Wrap
(24, 2, 1, 4.80, 1.30),   -- Latte
(24, 25, 1, 4.20, 1.30),  -- Apple Pie (price adjusted)
(25, 15, 1, 15.90, 5.30), -- Caesar Salad
(26, 10, 1, 18.90, 7.50), -- Big Breakfast
(26, 7, 1, 6.50, 2.10),   -- Green Smoothie
(27, 19, 1, 9.50, 2.90),  -- Bruschetta
(27, 1, 2, 4.50, 1.20),   -- 2 Cappuccinos
(28, 16, 1, 22.90, 9.20), -- Fish and Chips
(29, 12, 1, 14.20, 4.80), -- Pancakes
(29, 8, 1, 4.80, 1.60),   -- Orange Juice
(30, 18, 1, 8.90, 2.80),  -- Soup
(30, 20, 1, 7.80, 2.20),  -- Cheese Toastie
(30, 3, 1, 4.20, 1.15);   -- Flat White
```

- PAYMENT table

```sql
-- 8. PAYMENT Table
INSERT INTO PAYMENT (Order_ID, Payment_Amount, Payment_Method, Payment_Date, Payment_Status, Is_Refund) VALUES
(1, 23.40, 'Credit Card', '2024-09-18', 'Completed', FALSE),
(2, 9.30, 'Cash', '2024-09-18', 'Completed', FALSE),
(3, 31.80, 'Digital Wallet', '2024-09-18', 'Completed', FALSE),
(4, 18.90, 'EFTPOS', '2024-09-18', 'Completed', FALSE),
(5, 15.60, 'Credit Card', '2024-09-18', 'Completed', FALSE),
(6, 26.70, 'Cash', '2024-09-17', 'Completed', FALSE),
(7, 19.50, 'Digital Wallet', '2024-09-17', 'Completed', FALSE),
(8, 12.80, 'Debit Card', '2024-09-17', 'Completed', FALSE),
(9, 17.40, 'Credit Card', '2024-09-16', 'Completed', FALSE),
(10, 14.20, 'Cash', '2024-09-16', 'Completed', FALSE),
(11, 22.90, 'EFTPOS', '2024-09-15', 'Completed', FALSE),
(12, 28.40, 'Digital Wallet', '2024-09-15', 'Completed', FALSE),
(13, 8.90, 'Cash', '2024-09-14', 'Completed', FALSE),
(14, 33.50, 'Credit Card', '2024-09-14', 'Completed', FALSE),
(15, 16.70, 'Debit Card', '2024-09-13', 'Completed', FALSE),
(16, 25.30, 'Digital Wallet', '2024-09-10', 'Completed', FALSE),
(17, 21.60, 'Credit Card', '2024-09-08', 'Completed', FALSE),
(18, 13.40, 'Cash', '2024-09-05', 'Completed', FALSE),
(19, 29.80, 'EFTPOS', '2024-08-28', 'Completed', FALSE),
(20, 18.20, 'Debit Card', '2024-08-25', 'Completed', FALSE),
(24, 21.40, 'Credit Card', '2024-09-12', 'Completed', FALSE),
(25, 15.90, 'Cash', '2024-09-10', 'Completed', FALSE),
(26, 27.30, 'Digital Wallet', '2024-09-08', 'Completed', FALSE),
(27, 19.60, 'EFTPOS', '2024-09-06', 'Completed', FALSE),
(28, 23.80, 'Credit Card', '2024-09-04', 'Completed', FALSE),
(29, 16.50, 'Cash', '2024-09-02', 'Completed', FALSE),
(30, 22.10, 'Digital Wallet', '2024-08-30', 'Completed', FALSE);
```

- RESERVATION table

```sql
-- 9. RESERVATION Table
INSERT INTO RESERVATION (Customer_ID, Outlet_ID, Reservation_Date, Reservation_Time, Booked_Datetime, Party_Size, Status, Special_Requests) VALUES
(1, 1, '2024-09-20', '19:00:00', '2024-09-18 10:30:00', 2, 'Confirmed', 'Window table preferred'),
(3, 1, '2024-09-21', '12:30:00', '2024-09-18 14:15:00', 4, 'Confirmed', 'High chair needed'),
(5, 2, '2024-09-19', '18:30:00', '2024-09-17 16:45:00', 6, 'Confirmed', 'Birthday celebration'),
(7, 2, '2024-09-22', '11:00:00', '2024-09-18 09:20:00', 3, 'Confirmed', NULL),
(9, 3, '2024-09-20', '13:00:00', '2024-09-18 11:45:00', 2, 'Confirmed', 'Gluten-free options needed'),
(11, 1, '2024-09-23', '18:00:00', '2024-09-19 15:30:00', 5, 'Confirmed', 'Anniversary dinner'),
(13, 2, '2024-09-21', '14:30:00', '2024-09-18 13:20:00', 3, 'Confirmed', NULL),
(15, 3, '2024-09-24', '12:00:00', '2024-09-20 10:15:00', 8, 'Confirmed', 'Business lunch meeting'),
(17, 1, '2024-09-19', '20:00:00', '2024-09-17 14:30:00', 2, 'Completed', NULL),
(19, 2, '2024-09-18', '19:30:00', '2024-09-16 16:45:00', 4, 'Completed', 'Vegetarian preferences'),
(21, 3, '2024-09-17', '18:00:00', '2024-09-15 12:20:00', 6, 'Completed', 'Corporate event'),
(2, 1, '2024-09-16', '12:30:00', '2024-09-14 09:30:00', 2, 'No-show', NULL),
(4, 2, '2024-09-15', '19:00:00', '2024-09-13 17:45:00', 3, 'Cancelled', 'Family emergency'),
(6, 3, '2024-09-25', '17:30:00', '2024-09-20 14:15:00', 4, 'Confirmed', 'Outdoor seating preferred'),
(8, 1, '2024-09-26', '11:30:00', '2024-09-21 16:20:00', 2, 'Confirmed', 'Quiet table for proposal');
```

- SHIFT table

```sql
-- 10. SHIFT Table
INSERT INTO SHIFT (Outlet_ID, Shift_Date, Start_Time, End_Time, Shift_Type) VALUES
-- Sydney shifts
(1, '2024-09-18', '06:00:00', '14:00:00', 'Morning'),
(1, '2024-09-18', '14:00:00', '22:00:00', 'Evening'),
(1, '2024-09-17', '06:00:00', '14:00:00', 'Morning'),
(1, '2024-09-17', '14:00:00', '22:00:00', 'Evening'),
(1, '2024-09-16', '06:00:00', '14:00:00', 'Morning'),
(1, '2024-09-16', '14:00:00', '22:00:00', 'Evening'),
(1, '2024-09-15', '06:00:00', '14:00:00', 'Morning'),
-- Melbourne shifts
(2, '2024-09-18', '07:00:00', '15:00:00', 'Morning'),
(2, '2024-09-18', '15:00:00', '23:00:00', 'Evening'),
(2, '2024-09-17', '07:00:00', '15:00:00', 'Morning'),
(2, '2024-09-17', '15:00:00', '23:00:00', 'Evening'),
(2, '2024-09-16', '07:00:00', '15:00:00', 'Morning'),
(2, '2024-09-16', '15:00:00', '23:00:00', 'Evening'),
-- Brisbane shifts
(3, '2024-09-18', '06:30:00', '14:30:00', 'Morning'),
(3, '2024-09-18', '14:30:00', '22:30:00', 'Evening'),
(3, '2024-09-17', '06:30:00', '14:30:00', 'Morning'),
(3, '2024-09-17', '14:30:00', '22:30:00', 'Evening'),
(3, '2024-09-16', '06:30:00', '14:30:00', 'Morning'),
(3, '2024-09-16', '14:30:00', '22:30:00', 'Evening'),
(3, '2024-09-15', '06:30:00', '14:30:00', 'Morning');
```

- STAFF_SHIFT table

```sql
-- 11. STAFF_SHIFT Table
INSERT INTO STAFF_SHIFT (Staff_ID, Shift_ID, Check_In_Time, Check_Out_Time, Hours_Worked) VALUES
-- Sydney Morning Shifts
(1, 1, '2024-09-18 05:55:00', '2024-09-18 14:05:00', 8.17), -- Manager Tom
(2, 1, '2024-09-18 06:00:00', '2024-09-18 14:00:00', 8.00), -- Chef Maria
(3, 1, '2024-09-18 06:05:00', '2024-09-18 14:10:00', 8.08), -- Barista Jake
(4, 1, '2024-09-18 06:00:00', '2024-09-18 14:00:00', 8.00), -- Server Emma
(5, 1, '2024-09-18 06:10:00', '2024-09-18 14:15:00', 8.08), -- Kitchen Staff Alex

-- Sydney Evening Shifts
(1, 2, '2024-09-18 13:55:00', '2024-09-18 22:10:00', 8.25), -- Manager Tom
(6, 2, '2024-09-18 14:00:00', '2024-09-18 22:00:00', 8.00), -- Barista Sophie
(7, 2, '2024-09-18 14:05:00', '2024-09-18 22:05:00', 8.00), -- Server Ryan
(5, 2, '2024-09-18 14:00:00', '2024-09-18 22:15:00', 8.25), -- Kitchen Staff Alex

-- Sydney Previous Days
(1, 3, '2024-09-17 06:00:00', '2024-09-17 14:00:00', 8.00),
(2, 3, '2024-09-17 06:05:00', '2024-09-17 14:05:00', 8.00),
(3, 3, '2024-09-17 06:00:00', '2024-09-17 14:00:00', 8.00),
(4, 3, '2024-09-17 06:10:00', '2024-09-17 14:10:00', 8.00),

(6, 4, '2024-09-17 14:00:00', '2024-09-17 22:00:00', 8.00),
(7, 4, '2024-09-17 14:05:00', '2024-09-17 22:05:00', 8.00),
(5, 4, '2024-09-17 14:00:00', '2024-09-17 22:00:00', 8.00),

-- Melbourne Morning Shifts
(8, 8, '2024-09-18 06:55:00', '2024-09-18 15:05:00', 8.17), -- Manager Lisa
(9, 8, '2024-09-18 07:00:00', '2024-09-18 15:00:00', 8.00), -- Chef Marco
(10, 8, '2024-09-18 07:05:00', '2024-09-18 15:10:00', 8.08), -- Barista Chloe
(11, 8, '2024-09-18 07:00:00', '2024-09-18 15:00:00', 8.00), -- Server James
(12, 8, '2024-09-18 07:10:00', '2024-09-18 15:15:00', 8.08), -- Kitchen Staff Priya

-- Melbourne Evening Shifts
(8, 9, '2024-09-18 14:55:00', '2024-09-18 23:10:00', 8.25), -- Manager Lisa
(13, 9, '2024-09-18 15:00:00', '2024-09-18 23:00:00', 8.00), -- Barista Daniel
(11, 9, '2024-09-18 15:05:00', '2024-09-18 23:05:00', 8.00), -- Server James
(12, 9, '2024-09-18 15:00:00', '2024-09-18 23:15:00', 8.25), -- Kitchen Staff Priya

-- Brisbane Morning Shifts
(14, 14, '2024-09-18 06:25:00', '2024-09-18 14:35:00', 8.17), -- Manager Sarah
(15, 14, '2024-09-18 06:30:00', '2024-09-18 14:30:00', 8.00), -- Chef Tony
(16, 14, '2024-09-18 06:35:00', '2024-09-18 14:40:00', 8.08), -- Barista Jessica
(17, 14, '2024-09-18 06:30:00', '2024-09-18 14:30:00', 8.00), -- Server Michael
(18, 14, '2024-09-18 06:40:00', '2024-09-18 14:45:00', 8.08), -- Kitchen Staff Amy

-- Brisbane Evening Shifts
(14, 15, '2024-09-18 14:25:00', '2024-09-18 22:40:00', 8.25), -- Manager Sarah
(19, 15, '2024-09-18 14:30:00', '2024-09-18 22:30:00', 8.00), -- Server Chris
(20, 15, '2024-09-18 14:35:00', '2024-09-18 22:35:00', 8.00), -- Barista Zoe
(18, 15, '2024-09-18 14:30:00', '2024-09-18 22:45:00', 8.25), -- Kitchen Staff Amy

-- Previous day shifts for Melbourne
(8, 10, '2024-09-17 07:00:00', '2024-09-17 15:00:00', 8.00),
(9, 10, '2024-09-17 07:05:00', '2024-09-17 15:05:00', 8.00),
(10, 10, '2024-09-17 07:00:00', '2024-09-17 15:00:00', 8.00),
(11, 10, '2024-09-17 07:10:00', '2024-09-17 15:10:00', 8.00),

(13, 11, '2024-09-17 15:00:00', '2024-09-17 23:00:00', 8.00),
(11, 11, '2024-09-17 15:05:00', '2024-09-17 23:05:00', 8.00),
(12, 11, '2024-09-17 15:00:00', '2024-09-17 23:00:00', 8.00),

-- Brisbane previous shifts
(14, 16, '2024-09-17 06:30:00', '2024-09-17 14:30:00', 8.00),
(15, 16, '2024-09-17 06:35:00', '2024-09-17 14:35:00', 8.00),
(16, 16, '2024-09-17 06:30:00', '2024-09-17 14:30:00', 8.00),
(17, 16, '2024-09-17 06:40:00', '2024-09-17 14:40:00', 8.00);
```

Part B – Business Analytics Report

1.7 Business concern 1: Revenue Leakage from Reservation Management Failures

Name and Description: Quantifying Financial Impact of No-Shows and Cancellations on Peak-Time Revenue Optimization

Urban Eats loses revenue during its highest-margin trading windows because reservations fail. This analysis estimates the cost of no-shows and late cancellations by calculating lost revenue per seat in peak periods—12:00–14:00 (lunch) and 18:00–21:00 (dinner). Losses arise through three channels: (1) opportunity costs when walk-ins are turned away at capacity; (2) labour inefficiencies from staffing to forecast covers that do not materialise; and (3) inventory waste from pre-prepared items. Together, these factors explain the revenue leakage observed in peak service.

SQL Query

```sql
-- Financial impact of reservation failures (use outlet's real averages)
WITH
dinein AS (
  SELECT Outlet_ID, AVG(Total_Amount) AS avg_ticket
  FROM ORDERS
  WHERE Order_Type = 'Dine-in' AND Status = 'Completed'
  GROUP BY Outlet_ID
),
-- Avg party size for completed reservations per outlet (to get per-cover value)
party AS (
  SELECT Outlet_ID, AVG(Party_Size) AS avg_party_completed
  FROM RESERVATION
  WHERE Status = 'Completed'
  GROUP BY Outlet_ID
),
baseline AS (
  SELECT
    d.Outlet_ID,
    CASE
      WHEN COALESCE(p.avg_party_completed,0) > 0
        THEN d.avg_ticket / p.avg_party_completed
      ELSE 0
    END AS per_cover_value
  FROM dinein d
  LEFT JOIN party p ON p.Outlet_ID = d.Outlet_ID
)
SELECT
  r.Status AS Reservation_Status,
  o.Outlet_Name,
  COUNT(*)                          AS Count,
  AVG(r.Party_Size)                 AS Avg_Party_Size,
  SUM(r.Party_Size)                 AS Total_Seats_Lost,
  ROUND(SUM(r.Party_Size * COALESCE(b.per_cover_value,0)), 2) AS Estimated_Lost_Revenue,
  SUM(
    CASE
      WHEN (TIME(r.Reservation_Time) BETWEEN '12:00:00' AND '14:00:00')
        OR (TIME(r.Reservation_Time) BETWEEN '18:00:00' AND '21:00:00')
      THEN 1 ELSE 0
    END
  )                                 AS Peak_Time_Losses
FROM RESERVATION r
JOIN OUTLET o       ON o.Outlet_ID = r.Outlet_ID
LEFT JOIN baseline b ON b.Outlet_ID = r.Outlet_ID
WHERE r.Status IN ('No-show', 'Cancelled')
GROUP BY r.Status, o.Outlet_Name
ORDER BY Estimated_Lost_Revenue DESC;
```

SQL Result

Reservation_Stat...	Outlet_Name	Count	Avg_Party_Size	Total_Seats_Lost	Estimated_Lost_Revenue	Peak_Time_Losses
No-show	Urban Eats CBD Sydney	1	2.0000	2	23.75	1
Cancelled	Urban Eats South Melbourne	1	3.0000	3	17.45	1

Justification and Insights:

- Two failed reservations (five seats) produced **$41.20** in direct revenue loss, **100%** during peak windows (12:00–14:00; 18:00–21:00).
- **Sydney CBD no-show:** party of two, **$23.75** loss in the lunch rush, when rapid table turns drive throughput. Higher average spend per seat at this outlet amplifies the impact.
- **Melbourne cancellation:** party of three, **$17.45** loss during dinner service. Although margins are typically higher at dinner, lower per-seat spend at this outlet moderates the loss.

Strategic Impact: Losses are concentrated in the most profitable hours, indicating avoidable revenue leakage and reduced table utilisation.

Recommended actions

- **Tiered deposits: $10 per person** at peak times; **$5** off-peak.
- **24-hour confirmations:** automated call/SMS to reconfirm or cancel, improving accountability and freeing seats earlier.
- **Dynamic waitlist:** real-time backfill of cancelled tables to convert walk-ins and protect peak-period revenue.

1.8 Business concern 2: Customer Lifecycle Value Optimization and Churn Prevention

Name and Description: Strategic Customer Segmentation for Retention Investment and Lifetime Value Maximization

Urban Eats' customers are segmented by two factors: spend and recency. This matrix reveals three priority groups: high-spend but recently inactive customers (churn risk), one-time visitors (conversion focus), and loyal repeat customers (enhanced rewards). These patterns inform marketing budgets and facilitate targeted retention strategies.

SQL Query

```sql
SELECT
    c.Customer_ID,
    CONCAT(c.First_Name, ' ', c.Last_Name) AS Customer_Name,
    c.Registration_Date,
    c.Last_Visit_Date,
    COUNT(o.Order_ID) AS Total_Orders,
    ROUND(COALESCE(SUM(o.Total_Amount), 0), 2) AS Total_Spent,
    ROUND(COALESCE(AVG(o.Total_Amount), 0), 2) AS Average_Order_Value,
    DATEDIFF('2024-09-18', c.Last_Visit_Date) AS Days_Since_Last_Visit,
    CASE
        WHEN COUNT(o.Order_ID) >= 2 THEN 'Repeat Customer'
        WHEN COUNT(o.Order_ID) = 1 THEN 'One-Time Customer'
        ELSE 'Registered - No Orders'
    END AS Customer_Type,
    CASE
        WHEN COALESCE(SUM(o.Total_Amount), 0) >= 50 THEN 'High Value'
        WHEN COALESCE(SUM(o.Total_Amount), 0) >= 25 THEN 'Medium Value'
        WHEN COALESCE(SUM(o.Total_Amount), 0) > 0 THEN 'Low Value'
        ELSE 'No Purchases'
    END AS Value_Segment,
    CASE
        WHEN DATEDIFF('2024-09-18', c.Last_Visit_Date) <= 3 THEN 'Very Recent'
        WHEN DATEDIFF('2024-09-18', c.Last_Visit_Date) <= 7 THEN 'Recent'
        WHEN DATEDIFF('2024-09-18', c.Last_Visit_Date) <= 30 THEN 'Active'
        ELSE 'At Risk'
    END AS Recency_Status
FROM CUSTOMER c
LEFT JOIN ORDERS o ON c.Customer_ID = o.Customer_ID AND o.Status = 'Completed'
GROUP BY c.Customer_ID, c.First_Name, c.Last_Name, c.Registration_Date, c.Last_Visit_Date
ORDER BY Total_Spent DESC, Total_Orders DESC;
```

SQL Result

Customer_ID	Customer_Name	Registration_Da...	Last_Visit_Date	Total_Orders	Total_Spent	Average_Order_Value	Days_Since_Last_Vi...	Customer_Type	Value_Segment	Recency_S...
9	Jessica Thomas	2024-02-14	2024-09-09	4	82.40	20.60	9	Repeat Customer	High Value	Active
13	Rachel Martin	2024-03-22	2024-09-05	4	62.00	15.50	13	Repeat Customer	High Value	Active
12	Daniel Harris	2024-04-08	2024-09-07	2	56.80	28.40	11	Repeat Customer	High Value	Active
16	Benjamin Martinez	2024-01-08	2024-09-01	2	50.60	25.30	17	Repeat Customer	High Value	Active
17	Olivia Robinson	2024-04-25	2024-08-28	2	43.20	21.60	21	Repeat Customer	Medium Value	Active
4	James Brown	2024-03-05	2024-09-08	2	37.80	18.90	10	Repeat Customer	Medium Value	Active
15	Chloe Garcia	2024-05-15	2024-09-03	2	33.40	16.70	15	Repeat Customer	Medium Value	Active
8	Ryan Anderson	2024-03-18	2024-08-20	2	25.60	12.80	29	Repeat Customer	Medium Value	Active
2	Michael Chen	2024-02-20	2024-09-10	2	18.60	9.30	8	Repeat Customer	Low Value	Active
11	Amy White	2024-01-30	2024-07-15	4	78.80	19.70	65	Repeat Customer	High Value	At Risk
14	Andrew Thompson	2024-02-10	2024-06-30	2	67.00	33.50	80	Repeat Customer	High Value	At Risk
19	Isabella Rodriguez	2024-05-20	2024-08-10	2	59.60	29.80	39	Repeat Customer	High Value	At Risk
20	Joshua Lewis	2024-02-05	2024-07-25	2	36.40	18.20	55	Repeat Customer	Medium Value	At Risk
18	Nathan Clark	2024-03-12	2024-08-15	2	26.80	13.40	34	Repeat Customer	Medium Value	At Risk
24	Lucas Allen	2024-03-28	2024-08-05	0	0.00	0.00	44	Registered - No Orders	No Purchases	At Risk
3	Emma Williams	2024-01-10	2024-09-12	4	95.40	23.85	6	Repeat Customer	High Value	Recent
5	Lisa Davis	2024-02-28	2024-09-14	4	85.80	21.45	4	Repeat Customer	High Value	Recent
7	Sophie Taylor	2024-04-12	2024-09-13	4	78.20	19.55	5	Repeat Customer	High Value	Recent
6	David Wilson	2024-01-25	2024-09-11	2	53.40	26.70	7	Repeat Customer	High Value	Recent
25	Zoe Young	2024-06-15	2024-09-14	0	0.00	0.00	4	Registered - No Orders	No Purchases	Recent
1	Sarah Johnson	2024-01-15	2024-09-15	4	89.60	22.40	3	Repeat Customer	High Value	Very Recent
10	Matthew Jackson	2024-05-03	2024-09-16	2	28.40	14.20	2	Repeat Customer	Medium Value	Very Recent
21	Grace Lee	2024-06-01	2024-09-17	0	0.00	0.00	1	Registered - No Orders	No Purchases	Very Recent
22	Ethan Walker	2024-04-30	2024-09-18	0	0.00	0.00	0	Registered - No Orders	No Purchases	Very Recent
23	Mia Hall	2024-05-10	2024-09-16	0	0.00	0.00	2	Registered - No Orders	No Purchases	Very Recent

Justification and Insights:

- Retention gap: Five high-value customers (≥ $50 spend) have not visited in 30+ days, putting ~$334 of revenue at risk.
- Retention potential: Emma Williams has spent $95.40 across four orders, illustrating the value that successful retention can unlock.
- Onboarding issue: Five customers registered but never ordered, indicating conversion and onboarding failures.

Strategic Impact: A structured retention program will raise customer lifetime value while lowering acquisition costs.

- **Segmented actions:**
  - *High-value repeaters:* introduce VIP benefits and early-access offers.
  - *At-risk high-value customers:* run win-back campaigns and reminders.
  - *Non-purchasers:* provide welcome incentives and guided onboarding.
- **Resource focus:** Prioritise retention spend for customers with **≥ $70** cumulative spend to maximise ROI and lifetime value.
- **Automation:** Trigger email/SMS based on visit recency, with offers scaled to customer value segment.

1.9 Business concern 3: Menu Engineering Through Cross-Selling Intelligence

Name and Description: Product Bundling Strategy to Maximize Average Order Value

Basket analysis shows which items customers routinely choose together. These natural pairings support two actions: (1) design menu bundles that mirror actual behaviour, and (2) train staff to cross-sell complementary items. Aligning Urban Eats' offers and scripts with these patterns makes bundles feel natural and keeps recommendations relevant to customer preferences.

SQL Query

```sql
SELECT
    p1.Product_ID AS Product1_ID,
    p1.Product_Name AS Product1_Name,
    p2.Product_ID AS Product2_ID,
    p2.Product_Name AS Product2_Name,
    COUNT(*) AS Times_Ordered_Together
FROM ORDER_ITEM oi1
JOIN ORDER_ITEM oi2 ON oi1.Order_ID = oi2.Order_ID
JOIN PRODUCT p1 ON oi1.Product_ID = p1.Product_ID
JOIN PRODUCT p2 ON oi2.Product_ID = p2.Product_ID
WHERE oi1.Product_ID < oi2.Product_ID
GROUP BY oi1.Product_ID, oi2.Product_ID
HAVING Times_Ordered_Together >= 2
ORDER BY Times_Ordered_Together DESC;
```

SQL Result

Product1_ID	Product1_Name	Product2_ID	Product2_Name	Times_Ordered_Together
1	Cappuccino	19	Bruschetta	2
6	Iced Coffee	14	Gourmet Burger	2
6	Iced Coffee	21	Chocolate Brownie	2
14	Gourmet Burger	21	Chocolate Brownie	2

Justification and Insights:

- Basket analysis highlights two frequent pairs: Cappuccino + Bruschetta (2 co-occurrences) and Gourmet Burger + Chocolate Brownie (2 co-occurrences).
- These pairings suggest two use-cases: coffee with a light Mediterranean snack, and a premium main followed by an indulgent dessert.

Strategic Impact:

- Create a **"Perfect Pairs"** menu section with bundled pricing based on these combinations.
- **Upsell scripts:** baristas suggest bruschetta with cappuccinos; waitstaff offer brownies with gourmet burgers.
- Measure impact via attach rate and average order value (AOV). This approach should raise AOV while improving satisfaction through offers that match observed preferences.

1.10 Business concern 4: Menu Profitability Optimization and Portfolio Rationalization

Name and Description: Strategic Menu Engineering Using Profit-Volume Matrix Analysis for Optimal Product Mix

Urban Eats needs data-driven menu optimisation to retire low-performers and scale high-margin sellers. This analysis applies the **Profit–Popularity Matrix** to classify items as **Stars** (high profit, high popularity), **Workhorses** (low profit, high popularity), **Puzzles** (high profit, low popularity), and **Underperformers** (low profit, low popularity). Each group requires tailored actions—pricing/portion changes, targeted promotion, or retirement—to maximise menu profitability.

SQL Query

```sql
SELECT
    p.Product_ID,
    p.Product_Name,
    pc.Category_Name,
    COUNT(oi.Product_ID) AS Times_Ordered,
    SUM(oi.Quantity) AS Total_Units_Sold,
    ROUND(SUM(oi.Quantity * oi.Unit_Price), 2) AS Total_Revenue,
    ROUND(SUM(oi.Quantity * (oi.Unit_Price - p.Cost)), 2) AS Total_Profit,
    ROUND(
        (SUM(oi.Quantity * (oi.Unit_Price - p.Cost)) / SUM(oi.Quantity * oi.Unit_Price)) * 100,
        2
    ) AS Profit_Margin_Percent
FROM PRODUCT p
JOIN PRODUCT_CATEGORY pc ON p.Category_ID = pc.Category_ID
LEFT JOIN ORDER_ITEM oi ON p.Product_ID = oi.Product_ID
GROUP BY p.Product_ID, p.Product_Name, pc.Category_Name
ORDER BY Total_Profit DESC;
```

SQL Result

Product_ID	Product_Name	Category_Name	Times_Ordered	Total_Units_So...	Total_Revenue	Total_Profit	Profit_Margin_Percent
14	Gourmet Burger	Lunch Mains	3	3	58.50	35.10	60.00
10	Big Breakfast	Breakfast Items	3	3	56.70	34.20	60.32
15	Caesar Salad	Lunch Mains	3	3	47.70	31.80	66.67
12	Pancakes Stack	Breakfast Items	3	3	42.60	28.20	66.20
16	Fish and Chips	Lunch Mains	2	2	45.80	27.40	59.83
1	Cappuccino	Hot Beverages	5	8	36.00	26.40	73.33
21	Chocolate Brownie	Desserts	3	4	35.60	25.20	70.79
17	Chicken Wrap	Lunch Mains	2	2	27.60	17.80	64.49
11	Avocado Toast	Breakfast Items	2	2	25.00	16.60	66.40
13	Breakfast Burrito	Breakfast Items	2	2	23.60	15.40	65.25
2	Latte	Hot Beverages	4	4	19.20	14.00	72.92
19	Bruschetta	Light Meals	2	2	19.00	13.20	69.47
18	Soup of the Day	Light Meals	2	2	17.80	12.20	68.54
20	Cheese Toastie	Light Meals	2	2	15.60	11.20	71.79
29	Vegan Burger	Vegan Options	1	1	17.90	11.10	62.01
30	Coconut Curry	Vegan Options	1	1	16.50	10.60	64.24
28	Quinoa Salad	Healthy Options	1	1	14.50	9.30	64.14
6	Iced Coffee	Cold Beverages	3	3	12.60	8.85	70.24
7	Green Smoothie	Cold Beverages	2	2	13.00	8.80	67.69
27	Acai Bowl	Healthy Options	1	1	12.90	8.40	65.12
22	Cheesecake Slice	Desserts	1	1	9.50	6.40	67.37
8	Fresh Orange Juice	Cold Beverages	2	2	9.60	6.40	66.67
3	Flat White	Hot Beverages	2	2	8.40	6.10	72.62
24	Croissant	Pastries	1	1	4.80	3.50	72.92
26	Danish Pastry	Pastries	1	1	5.20	3.50	67.31
4	Hot Chocolate	Hot Beverages	1	1	5.20	3.40	65.38
5	Earl Grey Tea	Hot Beverages	1	1	3.80	2.90	76.32
25	Blueberry Muffin	Pastries	1	1	4.20	2.70	64.29
9	Iced Tea	Cold Beverages	1	1	3.50	2.55	72.86
23	Apple Pie	Desserts	1	1	4.20	1.80	42.86
31	Pumpkin Spice L...	Seasonal Spec...	0	NULL	NULL	NULL	NULL

Justification and Insights:

- **Stars:** *Gourmet Burger* and *Big Breakfast* deliver **$35.10** and **$34.20** profit respectively, with **3 orders each** and **≥60%** margins—high popularity and contribution.
- **Underperformer:** *Apple Pie* shows a **42.86%** margin and very low volume, pointing to pricing, portion, or recipe issues.
- **Puzzles:** High-margin, low-volume items warrant testing to uncover demand barriers (price perception, placement, naming).

Strategic Impact:

- **Feature Stars** prominently (menu position, social proof) and train staff to recommend them.
- **Fix Puzzles** via A/B tests on price/portion, bundle with natural complements, and trial **seasonal promotions**.
- **Remediate or retire Underperformers** through cost re-engineering or removal if contribution remains low.
- **Track impact** using mix shift and item contribution margin to confirm uplift in overall profitability.

1.11 Business concern 5: Labor Cost Control Through Attendance Pattern Analysis and Time Theft Prevention

Name and Description: Systematic Detection of Time Irregularities and Labor Cost Leakage Through Shift Pattern Analysis

We analyse time-clock records to detect late arrivals, early departures, and irregular entries that may signal time manipulation. Variances between scheduled and actual hours are

quantified and translated into wage impact. Findings will inform targeted actions, including coaching, roster adjustments, and tighter time-keeping controls, to reduce labour-cost leakage while maintaining fair employment practices.

SQL Query

```sql
-- Time irregularity and potential time theft analysis
SELECT
    CONCAT(s.First_Name, ' ', s.Last_Name) AS Staff_Name,
    s.Role,
    o.Outlet_Name,
    sh.Shift_Date,
    sh.Start_Time AS Scheduled_Start,
    sh.End_Time AS Scheduled_End,
    TIME(ss.Check_In_Time) AS Actual_Check_In,
    TIME(ss.Check_Out_Time) AS Actual_Check_Out,
    -- Calculate time discrepancies
    ROUND(TIME_TO_SEC(TIMEDIFF(TIME(ss.Check_In_Time), sh.Start_Time)) / 60.0, 1) AS Minutes_Late,
    ROUND(TIME_TO_SEC(TIMEDIFF(sh.End_Time, TIME(ss.Check_Out_Time))) / 60.0, 1) AS Minutes_Early_Leave,
    ss.Hours_Worked,
    ROUND(TIME_TO_SEC(TIMEDIFF(sh.End_Time, sh.Start_Time)) / 3600.0, 2) AS Scheduled_Hours,
    ROUND(ss.Hours_Worked - TIME_TO_SEC(TIMEDIFF(sh.End_Time, sh.Start_Time)) / 3600.0, 2) AS Hours_Variance,
    ROUND((ss.Hours_Worked - TIME_TO_SEC(TIMEDIFF(sh.End_Time, sh.Start_Time)) / 3600.0) * s.Hourly_Rate, 2) AS Wage_Impact,
    CASE
        WHEN TIME_TO_SEC(TIMEDIFF(TIME(ss.Check_In_Time), sh.Start_Time)) > 300 THEN 'CHRONIC LATENESS'
        WHEN TIME_TO_SEC(TIMEDIFF(sh.End_Time, TIME(ss.Check_Out_Time))) > 300 THEN 'EARLY DEPARTURE'
        WHEN ss.Hours_Worked > TIME_TO_SEC(TIMEDIFF(sh.End_Time, sh.Start_Time)) / 3600.0 + 0.5 THEN 'OVERTIME CONCERN'
        ELSE 'NORMAL'
    END AS Time_Flag
FROM STAFF_SHIFT ss
JOIN STAFF s ON ss.Staff_ID = s.Staff_ID
JOIN SHIFT sh ON ss.Shift_ID = sh.Shift_ID
JOIN OUTLET o ON s.Home_Outlet_ID = o.Outlet_ID
WHERE ss.Check_In_Time IS NOT NULL AND ss.Check_Out_Time IS NOT NULL
HAVING Time_Flag != 'NORMAL'
ORDER BY ABS(Wage_Impact) DESC;
```

SQL Result

Staff_Name	Role	Outlet_Name	Shift_Date	Scheduled_Start	Scheduled_End	Actual_Check_In	Actual_Check_Out	Minutes_Late	Minutes_Early_Leave	Hours_Worked	Scheduled_Hours	Hours_Variance	Wage_Impact	Time_Flag
Emma Kelly	Server	Urban Eats CBD Sydney	2024-09-17	06:00:00	14:00:00	06:10:00	14:10:00	10.0	-10.0	8.00	8.00	0.00	0.00	CHRONIC LATENESS
James Stewart	Server	Urban Eats South Melbourne	2024-09-17	07:00:00	15:00:00	07:10:00	15:10:00	10.0	-10.0	8.00	8.00	0.00	0.00	CHRONIC LATENESS
Michael Fraser	Server	Urban Eats Brisbane City	2024-09-17	06:30:00	14:30:00	06:40:00	14:40:00	10.0	-10.0	8.00	8.00	0.00	0.00	CHRONIC LATENESS
Priya Patel	Kitchen Staff	Urban Eats South Melbourne	2024-09-18	07:00:00	15:00:00	07:10:00	15:15:00	10.0	-15.0	8.08	8.00	0.08	1.88	CHRONIC LATENESS
Alex Nguyen	Kitchen Staff	Urban Eats CBD Sydney	2024-09-18	06:00:00	14:00:00	06:10:00	14:15:00	10.0	-15.0	8.08	8.00	0.08	1.84	CHRONIC LATENESS
Amy Davis	Kitchen Staff	Urban Eats Brisbane City	2024-09-18	06:30:00	14:30:00	06:40:00	14:45:00	10.0	-15.0	8.08	8.00	0.08	1.84	CHRONIC LATENESS

Justification and Insights:

- **Tardiness pattern:** Six staff repeatedly arrive **≥10 minutes** late, generating wage overpayments of **$1.84–$1.88** per occurrence.
- **Variance by role:** Kitchen shifts show the largest discrepancies, with some logs **0.08 hours (~5 min)** beyond schedule.
- **Cost impact:** While each event is small, the cumulative effect across staff and weeks produces meaningful annual leakage.

Strategic Impact:

- **Policy**: Introduce a progressive tardiness policy with a **5-minute grace window**; require approval for any overtime beyond rostered hours.
- **Controls**: Implement digital time-tracking with GPS/geofencing and automatic variance alerts.
- **Management**: Train supervisors to intervene early; adjust rosters for chronically late staff (later start times) to maintain coverage.

- **Monitoring**: Track late-arrival rate, unauthorised-overtime minutes, and wage variance to verify cost reductions and accountability.

1.12 Conclusion

This analytics and database design report has systematically identified and quantified several critical operational and financial challenges facing Urban Eats Café Chain. Through the development of a robust relational database and targeted SQL analysis, we have transformed raw data into actionable strategic insights.

The key findings reveal significant opportunities for improvement across five core business functions:

1. **Revenue Protection:** Quantifying the direct financial impact of reservation no-shows during peak hours highlights an easily addressable source of revenue leakage.
2. **Customer Retention:** Segmenting the customer base by value and recency has identified high-risk churn candidates and untapped potential, enabling precise and cost-effective retention marketing.
3. **Sales Growth:** Basket analysis has uncovered natural product affinities, providing a data-driven foundation for effective bundling and staff upsell training to increase the average order value.
4. **Profitability Optimization:** Classifying the menu based on profit and popularity has created a clear strategic roadmap to promote high-margin stars, remediate underperformers, and rationalize the product portfolio.
5. **Cost Control:** Analysing shift attendance patterns has exposed irregularities in labour scheduling, presenting an opportunity to tighten controls and reduce wage leakage.

The interconnected nature of these findings underscores that Urban Eats's path to enhanced profitability lies not in isolated fixes, but in a holistic, data-informed strategy. The implemented database serves as the single source of truth, enabling continuous monitoring of these KPIs.