Business Rules — Mamma Mia's Pizza

(Week 1)

Goal (Week 1): capture the business rules that drive our ERD and schema decisions. Pizza prices are **computed** (not stored), orders must include pizzas, delivery is assigned by postal code with a cooldown, and discounts are applied in a strict order.

---

1) Pricing (computed, not stored)

- We **do not** store a `price` on `pizzas`. Price is computed at query time from ingredients.
- **Units standard:**
  - `ingredients.cost_per_unit` is stored as **€/kg**.
  - `pizza_ingredients.qty_unit` is stored in **grams**.
  - During computation we convert grams → kg: `(qty_unit / 1000.0) * cost_per_unit`.
- **Formula per size**
  `price(size) = Σ( (qty_g / 1000) × ingredient_cost_€/kg ) × 1.40 (margin) × 1.09 (VAT) × size_multiplier`
- **Size multipliers:** S = 0.80, M = 1.00, L = 1.30.
- **Rounding:** customer-facing prices are rounded to **€0.01 at the very end** (after margin & VAT).
- **Cheapest pizza (for birthday):** pick the pizza line with the **lowest computed size price** before any discounts; if tied, use the first in the order.
- **Cheapest drink (for birthday):** lowest `unit_price` among drinks on the order.

---

2) Menu & dietary labels

- **Vegetarian:** pizza contains **no ingredient** where `is_meat = 1`.
- **Vegan:** pizza contains **no ingredient** where `is_meat = 1` **and** no ingredient where `is_animal_product = 1` (e.g., cheese, eggs).
- Drinks and desserts store their own `unit_price` and a `vegan` flag.
- Pizzas have an `active` flag so we can hide items seasonally without deleting.

---

3) Customers & addresses (data quality)

- `customers.email` is **UNIQUE**.

- `birth_date` must indicate **age ≥ 10** at time of creation (checked by constraint or application).
- A customer can have **many** `addresses` (1→N). Each `order` references **exactly one** delivery `address`.
- **Reporting fields:**
  - Age is **derived** from `birth_date` at query time.
  - To support "earnings by gender," we may include an **optional** `gender` column (e.g., `ENUM('F','M','X','PreferNotToSay')`) in a later iteration; queries should handle `NULL`.

---

4) Orders & order items

- An order must contain **at least one pizza**; drinks/desserts are optional.
- `placed_at` is set on creation.
- **Status lifecycle (allowed transitions):**
  `PLACED → PREPARING → OUT_FOR_DELIVERY → DELIVERED` (forward only).
  `CANCELLED` is allowed **only** from `PLACED` or `PREPARING`.
- Order line `quantity` must be **> 0**. Pizza lines also store a **size** (`S`,`M`,`L`).

---

5) Discounts (strict stacking order)

Apply discounts **in this exact order**:

1. **Birthday (calendar day match, server local date):**
   - 1 **cheapest pizza free** (see rule above)
   - **+ 1 drink free** (cheapest drink on the order)
2. **Loyalty:** once a customer has purchased **≥ 10 pizzas** (lifetime), apply **10%** off the **remaining pizza subtotal**.
3. **Discount code:** `discount_codes.code` is unique. If `single_use = 1`, it can be redeemed **once globally**. On redemption we set `redeemed_at` and `redeemed_by_customer`. Apply `percent_off` to the **post-loyalty subtotal**.

- The **final total cannot be negative**.
- If multiple discounts could hit the same item, earlier rules win; later rules apply to whatever subtotal remains.

---

6) Delivery & assignment

- Each `delivery_person` covers one or more postal codes via `delivery_zones` (1→N).
- Each order has exactly **one** row in `deliveries` (PK = `order_id`), with `assigned_to` → `delivery_people.delivery_person_id`.
- **Availability rule:** a driver is available if they have **no undelivered orders** (`deliveries.delivered_at IS NULL` none exist) **and** their most recent `delivered_at` is **≥ 30 minutes** ago.
- If no driver covers the customer's postal code (or no one is available), the order cannot move to `OUT_FOR_DELIVERY`.

---

7) Integrity constraints (schema + logic)

- `ingredients.cost_per_unit > 0`.
- `pizza_ingredients.qty_unit > 0`; all order-line `quantity > 0`.
- `discount_codes.code` is **UNIQUE**; `customers.email` is **UNIQUE**.
- `orders.status` and `order_pizzas.size` use **ENUM**s.
- Vegetarian/vegan labels are **derived** from ingredients (no redundant flags on `pizzas`).

---

8) Foreign-key behavior (matches the ERD)

- `addresses.customer_id → customers.customer_id`: **ON UPDATE CASCADE**, **ON DELETE CASCADE**.
- `orders.customer_id → customers.customer_id`: **ON UPDATE CASCADE**, **ON DELETE RESTRICT**.
- `orders.address_id → addresses.address_id`: **ON UPDATE CASCADE**, **ON DELETE RESTRICT**.
- `orders.discount_code_id → discount_codes.discount_code_id`: **ON UPDATE CASCADE**, **ON DELETE SET NULL**.
- `pizza_ingredients.pizza_id → pizzas.pizza_id`: **ON UPDATE CASCADE**, **ON DELETE CASCADE**.
- `pizza_ingredients.ingredient_id → ingredients.ingredient_id`: **ON UPDATE CASCADE**, **ON DELETE RESTRICT**.
- `order_pizzas.order_id → orders.order_id`: **ON UPDATE CASCADE**, **ON DELETE CASCADE**.
- `order_pizzas.pizza_id → pizzas.pizza_id`: **ON UPDATE CASCADE**, **ON DELETE RESTRICT**.
- `order_drinks.order_id → orders.order_id`: **ON UPDATE CASCADE**, **ON DELETE CASCADE**.
- `order_drinks.drink_id → drinks.drink_id`: **ON UPDATE CASCADE**, **ON DELETE RESTRICT**.
- `order_desserts.order_id → orders.order_id`: **ON UPDATE CASCADE**, **ON DELETE CASCADE**.
- `order_desserts.dessert_id → desserts.dessert_id`: **ON UPDATE CASCADE**, **ON DELETE RESTRICT**.
- `discount_codes.redeemed_by_customer → customers.customer_id`: **ON UPDATE CASCADE**, **ON DELETE SET NULL**.
- `delivery_zones.delivery_person_id → delivery_people.delivery_person_id`: **ON UPDATE CASCADE**, **ON DELETE CASCADE**.
- `deliveries.order_id → orders.order_id`: **ON UPDATE CASCADE**, **ON DELETE CASCADE**.
- `deliveries.assigned_to → delivery_people.delivery_person_id`: **ON UPDATE CASCADE**, **ON DELETE RESTRICT**.

---

9) Transactions (preview for later weeks)

- **Place-order is atomic.** We insert the order header and all items, compute prices, apply discounts (in order), optionally mark a discount code as redeemed, and assign a driver. If **any** step fails (invalid code, no available

driver, constraint violation), we **ROLLBACK** and nothing is saved.

---

10) Assumptions

- Currency is **EUR**; VAT is **9%**; margin is **40%** (configurable later).
- Birthday check uses server **local** date (month/day match).
- Timestamps are stored as `DATETIME` (UTC handling can be added later).