Purpose

A flat, line-item dataset of restaurant transactions across 5 stores. One row = one item in an order. Orders are identified by a shared order_id across their items.

File format

Single CSV with a header row.

Quoting: only text fields are quoted ("). Numbers, dates and booleans are unquoted.

Currency: prices are numeric, no £ sign.

Date range: chosen at runtime (inclusive start, exclusive end).

Weekend flag: true for Friday, Saturday, Sunday.

Columns (order shown)

Column	**Type (Postgres)**	**Meaning / Rules**
order_id	uuid	Same value across all items of one order. Refunds use a **new** order_id.
order_line_no	smallint	1..N within the order.
store_id	smallint	1..5. Natural volume differences by store are pre-baked.
order_dt	timestamptz	Timestamp of the order (local "restaurant time"). Peak windows: 12–14 and 18–22.
dow	smallint	ISO day of week, **1=Mon … 7=Sun** (derived from order_dt).
is_weekend	boolean	true for **Fri/Sat/Sun**, else false.
channel	text	One of: Dine-in, Collection, Deliveroo, Uber Eats, Just Eat (random mix).
payment_method	text	Cash/Card for Dine-in & Collection; Online for delivery apps.

Column	Type (Postgres)	Meaning / Rules
customer_id	integer | null	Present randomly for Dine-in/Collection; **always null** for delivery apps.
staff_id	integer | null	Random **1–20** for Dine-in/Collection; **null** for delivery apps.
holiday_tag	text | null	Set on special dates (e.g. GoodFriday, EidAlFitr, SummerLow). Otherwise null.
is_refund	boolean	false for normal sales, true for refund rows (see below).
item_sku	text	Stable SKU, e.g. MA_BUTTER_CHIK. Portions/variants are separate SKUs.
item_name	text	Human-readable name.
category	text	One of: Starter, Main, Side, Bread, Drink, Dessert, SetMeal.
unit_price_gbp	numeric(8,2)	Fixed menu price at time of sale.
quantity	smallint	Usually **1** for Mains/Starters/Desserts/SetMeals; **1–6** for Sides/Breads/Drinks.
line_total_gbp	numeric(10,2)	unit_price_gbp × quantity. **Negative** on refund rows.
promo_code	text | null	One of 10OFF,15OFF,20OFF at order level; only set if pre-discount order total ≥ £25. (No discount amount stored.)

Behavioural rules (generation logic)

Order size: each order has 2–15 items (random), forced to include ≥1 Main and ≥1 Side. Mains/Sides are favoured overall.

Volumes: Weekdays (Mon–Thu) ~80–100 orders/store/day; Weekends (Fri–Sun) ~120–200. Store-level multipliers create busier/quieter branches.

Special days: Hard-coded tags (2023–2026). Spikes (multiplier=1.3): UK bank holidays, Xmas period, NYE. Dips (0.8): Eid (approx), summer lulls. holiday_tag is filled on those dates.

Promotions: ~5–10% of orders receive a promo code if their pre-discount order total ≥ £25. No cap; value not stored (recompute later if needed).

Refunds: ~1% chance per order (monthly average). Refund is modelled by duplicating all original lines onto a later timestamp (same month) with is_refund=true and negative line_total_gbp. A new order_id is used for the refund document.

Channels & payment: per rules above; delivery apps always Online.

IDs: order_id is a UUID; customer_id appears only for Dine-in/Collection (sometimes); staff_id only for Dine-in/Collection.