```python
###### BASE MODELS #####
class MenuPlan(Base):
    __tablename__ = "menu_plan"
    # Composite PK: Date and Meal (assuming one entry per meal per day)
    date: Mapped[datetime] = mapped_column(Date, primary_key=True)
    meal: Mapped[str] = mapped_column(String(200), ForeignKey("recipes.name"),
primary_key=True)
    portions: Mapped[float] = mapped_column(Float)

class Diet(Base):
    __tablename__ = "diets"
    diet: Mapped[str] = mapped_column(String(100), primary_key=True)
    problematic_component: Mapped[str] = mapped_column(String(100),
primary_key=True)

class Preference(Base):
    __tablename__ = "preferences"
    # FK to Ingredients name
    article: Mapped[str] = mapped_column( String(100), primary_key=True )
    problematic_component: Mapped[str] = mapped_column(String(100),
primary_key=True)




class Nutrient(Base):
    __tablename__ = "nutrients"
    # Primary Key
    ingredient_group: Mapped[str] = mapped_column(String(100), primary_key=True)
    # Nutritional values (using Float for all numeric data)
    kcal: Mapped[float] = mapped_column(Float)
    kj: Mapped[float] = mapped_column(Float)
    fat: Mapped[float] = mapped_column(Float)
    saturated_fatty_acids: Mapped[float] = mapped_column(Float)
    mono_unsaturated_fatty_acids: Mapped[float] = mapped_column(Float)
    polyunsaturated_fatty_acids: Mapped[float] = mapped_column(Float)
    cholesterol_mg: Mapped[float] = mapped_column(Float)
    carbohydrates: Mapped[float] = mapped_column(Float)
    sugar: Mapped[float] = mapped_column(Float)
    starch: Mapped[float] = mapped_column(Float)
    dietary_fibre: Mapped[float] = mapped_column(Float)
    protein: Mapped[float] = mapped_column(Float)
    salt: Mapped[float] = mapped_column(Float)

    # Relationship back to ingredients in this group
    ingredients: Mapped[list["Ingredient"]] =
relationship(back_populates="nutrient_info")
```

```python

class Ingredient(Base):
    __tablename__ = "ingredients"
    name: Mapped[str] = mapped_column(String(100), primary_key=True)
    # Foreign Key to Nutrients
    group: Mapped[str] = mapped_column(ForeignKey("nutrients.ingredient_group"))
    unit: Mapped[str] = mapped_column(String(50), primary_key=True)
    g_per_unit : Mapped[float] = mapped_column(Float)
    price_per_unit : Mapped[float] = mapped_column(Float)
    store: Mapped[str] = mapped_column(String(50))
    quantity_on_stock : Mapped[float] = mapped_column(Float, nullable=False)
    expiration_date: Mapped[datetime] = mapped_column(Date, nullable=True)

    # Relationships
    nutrient_info: Mapped["Nutrient"] = relationship(back_populates="ingredients")
    used_in_recipes: Mapped[list["Instruction"]] =
relationship(back_populates="ingredient_name")

class Recipe(Base):
    __tablename__ = "recipes"
    name: Mapped[str] = mapped_column(String(200), primary_key=True)
    description: Mapped[str] = mapped_column(String(300))
    portions: Mapped[int] = mapped_column(Integer)

    ingredients_list: Mapped[list["Instruction"]] = relationship(
        back_populates="recipe_name",
        cascade="all, delete-orphan"
    )

class Instruction(Base):
    __tablename__ = "instructions"

    recipe: Mapped[str] = mapped_column(ForeignKey("recipes.name"),
primary_key=True)
    # Note: We remove the individual ForeignKeys from these two lines...
    ingredient: Mapped[str] = mapped_column(String(100), primary_key=True)
    unit: Mapped[str] = mapped_column(String(50), primary_key=True)

    quantity: Mapped[float] = mapped_column(Float)
    preparation: Mapped[str] = mapped_column(String(200), nullable=True)

    # ...and define them as a single Constraint here:
    __table_args__ = (
        ForeignKeyConstraint(
            ["ingredient", "unit"],
            ["ingredients.name", "ingredients.unit"],
        ),
    )

    # Relationships
    recipe_name: Mapped["Recipe"] = relationship(back_populates="ingredients_list")
    ingredient_name: Mapped["Ingredient"] =
relationship(back_populates="used_in_recipes")
```

```
"""
CREATE OR REPLACE VIEW v_ingredients_nutrition AS
SELECT
    name                    AS name,
    i.group                 AS group,
    unit                    AS unit,
    g_per_unit              AS g_per_unit,
    price_per_unit          AS price_per_unit,

    ROUND( ((g_per_unit/100)*kcal)::numeric , 4)
                            AS unit_kcal,
    ROUND( ((g_per_unit/100)*kj)::numeric , 4)
                            AS unit_kj,
    ROUND( ((g_per_unit/100)*fat)::numeric , 4)
                            AS g_per_unit_fat,
    ROUND( ((g_per_unit/100)*saturated_fatty_acids)::numeric , 4)
                            AS g_per_unit_saturated_fatty_acids,
    ROUND( ((g_per_unit/100)*mono_unsaturated_fatty_acids)::numeric , 4)
                            AS g_per_unit_mono_unsaturated_fatty_acids,
    ROUND( ((g_per_unit/100)*polyunsaturated_fatty_acids)::numeric , 4)
                            AS g_per_unit_polyunsaturated_fatty_acids,
    ROUND( ((g_per_unit/100)*cholesterol_mg)::numeric , 4)
                            AS unit_cholesterol_mg,
    ROUND( ((g_per_unit/100)*carbohydrates)::numeric , 4)
                            AS g_per_unit_carbohydrates,
    ROUND( ((g_per_unit/100)*sugar)::numeric , 4)
                            AS g_per_unit_sugar,
    ROUND( ((g_per_unit/100)*starch)::numeric , 4)
                            AS g_per_unit_starch,
    ROUND( ((g_per_unit/100)*dietary_fibre)::numeric , 4)
                            AS g_per_unit_dietary_fibre,
    ROUND( ((g_per_unit/100)*protein)::numeric , 4)
                            AS g_per_unit_protein,
    ROUND( ((g_per_unit/100)*salt)::numeric , 4)
                            AS g_per_unit_salt

FROM ingredients i
JOIN nutrients n ON  i.group = n.ingredient_group
ORDER BY name;
```

```
CREATE OR REPLACE VIEW v_instructions AS
SELECT
    recipe                          AS recipe,
    ingredient                      AS ingredient,
    i.unit                          AS unit,
    m.quantity                      AS quantity,
    quantity * i.g_per_unit         AS quantity_in_grams,
    preparation                     AS preparation,
    ROUND(
        (quantity * i.g_per_unit * i.price_per_unit)::numeric
        , 3
    )
                                    AS price,
    price_per_unit                  AS price_per_unit,
    store                           AS store,
    quantity_on_stock               AS quantity_on_stock,
    expiration_date                 AS expiration_date
FROM instructions m
JOIN ingredients i ON
    i.name = m.ingredient
    AND i.unit = m.unit
;

CREATE OR REPLACE VIEW v_meal_price_per_portion AS
SELECT
    i.recipe            AS recipe,
    ROUND(
        SUM(price_per_unit *quantity/portions)::numeric
        ,2
    )                   AS price_per_portion,
    MIN(r.portions )    AS default_portions,
    COUNT(*)            AS Nbr_of_Ingredients,
    COUNT(*)            AS Nbr_of_Ingredients,

FROM recipes r
JOIN v_instructions i ON  r.name = i.recipe
GROUP BY i.recipe
ORDER BY recipe;
"""
```