

```
python
1 ##### BASE MODELS #####
2 class MenuPlan(Base):
3     __tablename__ = "menu_plan"
4     # Composite PK: Date and Meal (assuming one entry per meal per day)
5     date: Mapped[datetime] = mapped_column(Date, primary_key=True)
6     meal: Mapped[str] = mapped_column(String(200), ForeignKey("recipes.name")),
7     primary_key=True)
8     portions: Mapped[float] = mapped_column(Float)
9
10
11 class Diet(Base):
12     __tablename__ = "diets"
13     diet: Mapped[str] = mapped_column(String(100), primary_key=True)
14     problematic_component: Mapped[str] = mapped_column(String(100),
15     primary_key=True)
16
17 class Preference(Base):
18     __tablename__ = "preferences"
19     # FK to Ingredients name
20     article: Mapped[str] = mapped_column(String(100), primary_key=True)
21     problematic_component: Mapped[str] = mapped_column(String(100),
22     primary_key=True)
23
24
25
26 class Nutrient(Base):
27     __tablename__ = "nutrients"
28     # Primary Key
29     ingredient_group: Mapped[str] = mapped_column(String(100), primary_key=True)
30     # Nutritional values (using Float for all numeric data)
31     kcal: Mapped[float] = mapped_column(Float)
32     kj: Mapped[float] = mapped_column(Float)
33     fat: Mapped[float] = mapped_column(Float)
34     saturated_fatty_acids: Mapped[float] = mapped_column(Float)
35     mono_unsaturated_fatty_acids: Mapped[float] = mapped_column(Float)
36     polyunsaturated_fatty_acids: Mapped[float] = mapped_column(Float)
37     cholesterol_mg: Mapped[float] = mapped_column(Float)
38     carbohydrates: Mapped[float] = mapped_column(Float)
39     sugar: Mapped[float] = mapped_column(Float)
40     starch: Mapped[float] = mapped_column(Float)
41     dietary_fibre: Mapped[float] = mapped_column(Float)
42     protein: Mapped[float] = mapped_column(Float)
43     salt: Mapped[float] = mapped_column(Float)
44
45     # Relationship back to ingredients in this group
46     ingredients: Mapped[list["Ingredient"]] =
47     relationship(back_populates="nutrient_info")
48
49
50
51
52
53
```

```
54
55 class Ingredient(Base):
56     __tablename__ = "ingredients"
57     name: Mapped[str] = mapped_column(String(100), primary_key=True)
58     # Foreign Key to Nutrients
59     group: Mapped[str] = mapped_column(ForeignKey("nutrients.ingredient_group"))
60     unit: Mapped[str] = mapped_column(String(50), primary_key=True)
61     g_per_unit : Mapped[float] = mapped_column(Float)
62     price_per_unit : Mapped[float] = mapped_column(Float)
63     store: Mapped[str] = mapped_column(String(50))
64     quantity_on_stock : Mapped[float] = mapped_column(Float, nullable=False)
65     expiration_date: Mapped[datetime] = mapped_column(Date, nullable=True)
66
67     # Relationships
68     nutrient_info: Mapped["Nutrient"] = relationship(back_populates="ingredients")
69     used_in_recipes: Mapped[list["Instruction"]] =
70         relationship(back_populates="ingredient_name")
71
72 class Recipe(Base):
73     __tablename__ = "recipes"
74     name: Mapped[str] = mapped_column(String(200), primary_key=True)
75     description: Mapped[str] = mapped_column(String(300))
76     portions: Mapped[int] = mapped_column(Integer)
77
78     ingredients_list: Mapped[list["Instruction"]] = relationship(
79         back_populates="recipe_name",
80         cascade="all, delete-orphan"
81     )
82
83 class Instruction(Base):
84     __tablename__ = "instructions"
85
86     recipe: Mapped[str] = mapped_column(ForeignKey("recipes.name"),
87     primary_key=True)
88     # Note: We remove the individual ForeignKeys from these two lines...
89     ingredient: Mapped[str] = mapped_column(String(100), primary_key=True)
90     unit: Mapped[str] = mapped_column(String(50), primary_key=True)
91
92     quantity: Mapped[float] = mapped_column(Float)
93     preparation: Mapped[str] = mapped_column(String(200), nullable=True)
94
95     # ...and define them as a single Constraint here:
96     __table_args__ = (
97         ForeignKeyConstraint(
98             ["ingredient", "unit"],
99             ["ingredients.name", "ingredients.unit"],
100            ),
101        )
102
103     # Relationships
104     recipe_name: Mapped["Recipe"] = relationship(back_populates="ingredients_list")
105     ingredient_name: Mapped["Ingredient"] =
106         relationship(back_populates="used_in_recipes")
107
108
109
```

```
109  
110  
111      """  
112 CREATE OR REPLACE VIEW v_ingredients_nutrition AS  
113 SELECT  
114     name                      AS name,  
115     i.group                 AS group,  
116     unit                    AS unit,  
117     g_per_unit             AS g_per_unit,  
118     price_per_unit        AS price_per_unit,  
119  
120     ROUND( ((g_per_unit/100)*kcal)::numeric , 4)  
121                           AS unit_kcal,  
122     ROUND( ((g_per_unit/100)*kj)::numeric , 4)  
123                           AS unit_kj,  
124     ROUND( ((g_per_unit/100)*fat)::numeric , 4)  
125                           AS g_per_unit_fat,  
126     ROUND( ((g_per_unit/100)*saturated_fatty_acids)::numeric , 4)  
127                           AS g_per_unit_saturated_fatty_acids,  
128     ROUND( ((g_per_unit/100)*mono_unsaturated_fatty_acids)::numeric , 4)  
129                           AS g_per_unit_mono_unsaturated_fatty_acids,  
130     ROUND( ((g_per_unit/100)*polyunsaturated_fatty_acids)::numeric , 4)  
131                           AS g_per_unit_polyunsaturated_fatty_acids,  
132     ROUND( ((g_per_unit/100)*cholesterol_mg)::numeric , 4)  
133                           AS unit_cholesterol_mg,  
134     ROUND( ((g_per_unit/100)*carbohydrates)::numeric , 4)  
135                           AS g_per_unit_carbohydrates,  
136     ROUND( ((g_per_unit/100)*sugar)::numeric , 4)  
137                           AS g_per_unit_sugar,  
138     ROUND( ((g_per_unit/100)*starch)::numeric , 4)  
139                           AS g_per_unit_starch,  
140     ROUND( ((g_per_unit/100)*dietary_fibre)::numeric , 4)  
141                           AS g_per_unit_dietary_fibre,  
142     ROUND( ((g_per_unit/100)*protein)::numeric , 4)  
143                           AS g_per_unit_protein,  
144     ROUND( ((g_per_unit/100)*salt)::numeric , 4)  
145                           AS g_per_unit_salt  
146  
147 FROM ingredients i  
148 JOIN nutrients n ON i.group = n.ingredient_group  
149 ORDER BY name;  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164
```

```

165
166
167
168
169 CREATE OR REPLACE VIEW v_instructions AS
170 SELECT
171     recipe                      AS recipe,
172     ingredient                   AS ingredient,
173     i.unit                       AS unit,
174     m.quantity                   AS quantity,
175     quantity * i.g_per_unit    AS quantity_in_grams,
176     preparation                 AS preparation,
177     ROUND(
178         (quantity * i.g_per_unit * i.price_per_unit)::numeric
179         , 3
180     )
181                         AS price,
182     price_per_unit              AS price_per_unit,
183     store                        AS store,
184     quantity_on_stock           AS quantity_on_stock,
185     expiration_date             AS expiration_date
186 FROM instructions m
187 JOIN ingredients i ON
188     i.name = m.ingredient
189     AND i.unit = m.unit
190 ;
191
192 CREATE OR REPLACE VIEW v_meal_price_per_portion AS
193 SELECT
194     i.recipe                     AS recipe,
195     ROUND(
196         SUM(price_per_unit * quantity/portions)::numeric
197         ,2
198     )                           AS price_per_portion,
199     MIN(r.portions )            AS default_portions,
200     COUNT(*)                    AS Nbr_of_Ingredients,
201     COUNT(*)                    AS Nbr_of_Ingredients,
202
203 FROM recipes r
204 JOIN v_instructions i ON  r.name = i.recipe
205 GROUP BY i.recipe
206 ORDER BY recipe;
207 """

```