```
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import ForeignKeyConstraint, PrimaryKeyConstraint, CheckConstraint, Numeric
from sqlalchemy.dialects.postgresql import JSONB, NUMERIC
from sqlalchemy.ext.mutable import MutableList

db = SQLAlchemy()

class User(db.Model):
    __tablename__ = 'User'

    email = db.Column(db.String, primary_key=True, unique=True, nullable=False)
    name = db.Column(db.String, nullable=False)
    date_of_birth = db.Column(db.DateTime, nullable=True)
    street = db.Column(db.String, nullable=True)
    housenr = db.Column(db.Integer, nullable=True)
    postalcode = db.Column(db.String, nullable=True)  # Veranderd naar String voor compatibiliteit met inter
    city = db.Column(db.String, nullable=True)
    country = db.Column(db.String, nullable=True)
    telephonenr = db.Column(db.String, nullable=True)
    is_chef = db.Column(db.Boolean, default=False, nullable=False)  # Nodig om te onderscheiden tussen c
    preferences = db.Column(db.JSON, nullable=False, default=lambda: {
        "allergies": [],
        "favorite_origins": [],
        "favorite_ingredients": []
    })
    favorites = db.Column(MutableList.as_mutable(JSONB), default=[])
```

```
profile_picture = db.Column(db.Text(255), nullable=True)
chef_description = db.Column(db.String(2500), nullable=True)

# Relaties
recipes = db.relationship('Recipe', backref='chef', lazy=True)
consumer_transactions = db.relationship(
    'Transaction',
    backref='consumer',
    lazy=True,
    primaryjoin="User.email == Transaction.consumer_email"
)
chef_transactions = db.relationship(
    'Transaction',
    backref='chef_user',
    lazy=True,
    primaryjoin="User.email == Transaction.chef_email"
)

def set_preferences(self, favorite_ingredients=None, favorite_origins=None, allergies=None):
    """
    Update user preferences.
    """
    self.preferences = {
        "favorite_ingredients": favorite_ingredients or [],
        "favorite_origins": favorite_origins or [],
        "allergies": allergies or [],
```

```
    }

def get_preferences(self):
    """
    Retrieve user preferences.
    """
    return self.preferences or {}

def __repr__(self):
    return f"<User(email={self.email}, name={self.name}, is_chef={self.is_chef})>"

def add_to_favorites(self, recipename, chef_email):
    if not self.favorites:
        self.favorites = []
    favorite_entry = {"recipename": recipename, "chef_email": chef_email}
    if favorite_entry not in self.favorites:
        self.favorites.append(favorite_entry)

def remove_from_favorites(self, recipename, chef_email):
    if not self.favorites:
        self.favorites = []
    favorite_entry = {"recipename": recipename, "chef_email": chef_email}
    self.favorites = [fav for fav in self.favorites if fav != favorite_entry]

def is_favorite(self, recipename, chef_email):
    # Ensure favorites is initialized as an empty list if None
```

```
        if not self.favorites:
            self.favorites = []
        favorite_entry = {"recipename": recipename, "chef_email": chef_email}
        return favorite_entry in self.favorites

class Recipe(db.Model):
    __tablename__ = 'recipe'

    recipename = db.Column(db.String, nullable=False)
    chef_email = db.Column(db.String, db.ForeignKey('User.email', ondelete='CASCADE'), nullable=False)
    chef_name = db.Column(db.String, nullable=False)  # Toegevoegd om overeen te komen met SQL-sche
    description = db.Column(db.String, nullable=True)
    duration = db.Column(db.Integer, nullable=True)
    price = db.Column(NUMERIC(10, 2), nullable=True)  # Wijziging naar NUMERIC voor prijs met decimale
    ingredients = db.Column(JSONB, nullable=True)  # Blijft als JSONB zoals het oorspronkelijk was
    allergiesrec = db.Column(db.Text, nullable=True)
    image = db.Column(db.Text, nullable=True)
    origin = db.Column(db.String, nullable=True)  # Toegevoegd voor herkomst
    category = db.Column(db.String, nullable=True)  # Toegevoegd voor categorie
    preparation = db.Column(db.Text, nullable=True)  # Toegevoegd voor bereidingswijze

    __table_args__ = (
        PrimaryKeyConstraint('recipename', 'chef_email'),
    )

    def __repr__(self):
```

```
        return f"<Recipe(recipename={self.recipename}, chef_email={self.chef_email})>"



class Transaction(db.Model):
    __tablename__ = 'transaction'

    transactionid = db.Column(db.BigInteger, primary_key=True)
    transactiondate = db.Column(db.DateTime, nullable=False)
    price = db.Column(db.Numeric(10, 2), nullable=False)
    consumer_email = db.Column(db.String, db.ForeignKey('User.email', ondelete='SET NULL'), nullable=T
    chef_email = db.Column(db.String, db.ForeignKey('User.email', ondelete='SET NULL'), nullable=False)
    recipename = db.Column(db.String, nullable=False)

    __table_args__ = (
        db.ForeignKeyConstraint(
            ['recipename', 'chef_email'],
            ['recipe.recipename', 'recipe.chef_email'],
            ondelete='CASCADE'
        ),
    )

    def __repr__(self):
        return f"<Transaction(transactionid={self.transactionid}, recipename={self.recipename})>"
```

```
class Review(db.Model):
    __tablename__ = 'review'

    reviewid = db.Column(db.BigInteger, primary_key=True)
    comment = db.Column(db.Text, nullable=True)
    rating = db.Column(db.Integer, nullable=False)
    reviewdate = db.Column(db.DateTime, default=db.func.current_timestamp())
    consumer_email = db.Column(db.String, db.ForeignKey('User.email', ondelete='SET NULL'), nullable=F
    recipename = db.Column(db.String, nullable=False)
    chef_email = db.Column(db.String, nullable=False)
    transactionid = db.Column(db.BigInteger, db.ForeignKey('transaction.transactionid', ondelete='CASCAD

    __table_args__ = (
        db.ForeignKeyConstraint(
            ['recipename', 'chef_email'],
            ['recipe.recipename', 'recipe.chef_email'],
            ondelete='CASCADE'
        ),
        db.CheckConstraint('rating BETWEEN 1 AND 5', name='check_rating_range'),
    )

    def __repr__(self):
        return f"<Review(reviewid={self.reviewid}, rating={self.rating})>"
```

```
class Feedback(db.Model):
    __tablename__ = 'feedback'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50), nullable=False)
    email = db.Column(db.String(100), nullable=False)
    subject = db.Column(db.String(100), nullable=False)
    message = db.Column(db.Text, nullable=False)
    is_public = db.Column(db.Boolean, default=False)  # Whether the comment is public
    created_at = db.Column(db.DateTime, default=db.func.now())
```