```
class Nutrient:
    def __init__(self, name, unit, daily_value):
        self.name = name
        self.unit = unit
        self.daily_value = daily_value
    def _repr_(self):
        return f"{self.name} ({self.unit}) - Daily Value: {self.daily_value}{self.unit}"

class Food:
    def __init__(self, name):
        self.name = name
        self.nutrients = {}

    def add_nutrient(self, nutrient, amount):
        self.nutrients[nutrient.name] = {"amount": amount, "unit": nutrient.unit}

    def get_nutritional_info(self):
        return ', '.join([f"{nutrient_name}: {details['amount']}{details['unit']}"
                          for nutrient_name, details in self.nutrients.items()])

    def _repr_(self):
        return f"Food: {self.name} | Nutrients: {self.get_nutritional_info()}"

class Athlete:
    def __init__(self, athlete_id, name, sport, weight, height, age, activity_level):
        self.athlete_id = athlete_id
        self.name = name
        self.sport = sport
        self.weight = weight
        self.height = height
        self.age = age
        self.activity_level = activity_level
        self.daily_intake = []
    def track_nutritional_intake(self, food):
        self.daily_intake.append(food)

    def calculate_daily_nutrition(self):
        daily_totals = {}
        for food in self.daily_intake:
            for nutrient, details in food.nutrients.items():
                if nutrient not in daily_totals:
                    daily_totals[nutrient] = 0
                daily_totals[nutrient] += details['amount']
        return daily_totals

    def _repr_(self):
        return f"Athlete: {self.name} | Activity Level: {self.activity_level}"

class NutritionDatabase:
    def __init__(self):
        self.nutrients = {}
        self.foods = {}

    def create_nutrient(self, name, unit, daily_value):
        if name not in self.nutrients:
            nutrient = Nutrient(name, unit, daily_value)
            self.nutrients[name] = nutrient
        else:
            print(f"Nutrient {name} already exists.")

    def read_nutrient(self, name):
        return self.nutrients.get(name, f"Nutrient {name} not found.")

    def update_nutrient(self, name, unit=None, daily_value=None):
        if name in self.nutrients:
            nutrient = self.nutrients[name]
            if unit:
                nutrient.unit = unit
            if daily_value:
                nutrient.daily_value = daily_value
        else:
            print(f"Nutrient {name} not found.")

    def delete_nutrient(self, name):
        if name in self.nutrients:
            del self.nutrients[name]
        else:
            print(f"Nutrient {name} not found.")

    def create_food(self, food_name):
        if food_name not in self.foods:
            food = Food(food_name)
            self.foods[food_name] = food
        else:
            print(f"Food {food_name} already exists.")
    def read_food(self, food_name):
        return self.foods.get(food_name, f"Food {food_name} not found.")

    def update_food(self, food_name, nutrient, amount):
        if food_name in self.foods:
            self.foods[food_name].add_nutrient(nutrient, amount)
        else:
            print(f"Food {food_name} not found.")

    def delete_food(self, food_name):
        if food_name in self.foods:
            del self.foods[food_name]
        else:
            print(f"Food {food_name} not found.")

class AthleteManager:
    def __init__(self):
        self.athletes = {}

    def add_athlete(self, athlete_id, name, sport, weight, height, age, activity_level):
        if athlete_id not in self.athletes:
            athlete = Athlete(athlete_id, name, sport, weight, height, age, activity_level)
            self.athletes[athlete_id] = athlete
        else:
            print(f"Athlete with ID {athlete_id} already exists.")

    def track_nutritional_intake(self, athlete_id, food):
        athlete = self.athletes.get(athlete_id)
        if athlete:
            athlete.track_nutritional_intake(food)
        else:
            print(f"Athlete with ID {athlete_id} not found.")

    def provide_dietary_guidance(self, athlete_id):
        athlete = self.athletes.get(athlete_id)
        if athlete:
            if athlete.activity_level == "high":
                return f"{athlete.name}'s Dietary Guidance: High-protein, high-carbohydrate diet
recommended."
            elif athlete.activity_level == "medium":
                return f"{athlete.name}'s Dietary Guidance: Balanced diet recommended."
            else:
                return f"{athlete.name}'s Dietary Guidance: Focus on low-calorie, nutrient-dense
```

```
foods."
        return "Athlete not found."

    def display_athlete_intake(self, athlete_id):
        athlete = self.athletes.get(athlete_id)
        if athlete:
            daily_totals = athlete.calculate_daily_nutrition()
            return ', '.join([f"{nutrient}: {amount}" for nutrient, amount in daily_totals.items()])
        return "Athlete not found."

if __name__ == "__main__":

    nutrition_db = NutritionDatabase()
    athlete_manager = AthleteManager()

    nutrition_db.create_nutrient("Protein", "g", 50)
    nutrition_db.create_nutrient("Carbohydrates", "g", 300)
    nutrition_db.create_nutrient("Fat", "g", 70)

    nutrition_db.create_food("Chicken Breast")
    nutrition_db.update_food("Chicken Breast", nutrition_db.read_nutrient("Protein"), 30)

    nutrition_db.create_food("Rice")
    nutrition_db.update_food("Rice", nutrition_db.read_nutrient("Carbohydrates"), 45)

    athlete_manager.add_athlete(1, "John Doe", "weightlifting", 75, 180, 25, "high")
    athlete_manager.add_athlete(2, "Jane Smith", "marathonrunning", 60, 165, 30, "medium")

    athlete_manager.track_nutritional_intake(1, nutrition_db.read_food("Chicken Breast"))
    athlete_manager.track_nutritional_intake(1, nutrition_db.read_food("Rice"))

    print(athlete_manager.provide_dietary_guidance(1))
    print(athlete_manager.provide_dietary_guidance(2))
    print(athlete_manager.display_athlete_intake(1))
```