In [12]:
```
from gurobipy import *
import csv

##### Single Run Version ##########

#---- One week example -----------#

# ---------------------------------
# Parameters
# ----------------------------------

# Number of available vaccines for the week
V = 150000

# Number of available staff for the week (V-vaccinating, A-administrative)
E_V = 205
E_A = 789

# Initial vaccine coverage C[i] in each county (as proportion vaccinated)
C = {
    "Franklin": 0.18,
    "Delaware": 0.09,
    "Licking": 0.0,
    "Fairfield": 0.0,
    "Union": 0.0,
    "Marion": 0.0,
    "Knox": 0.0,
    "Pickaway": 0.0,
    "Logan": 0.0,
    "Madison": 0.09,
    "Crawford": 0.18,
    "Morrow": 0.18,
    "Hardin": 0.18,
    "Fayette": 0.18,
    "Wyandot": 0.18
}

# Maximum available clinic square footage S[i] for each county
S = {
    "Franklin": 452082,
    "Delaware": 123766,
    "Licking": 27336,
    "Fairfield": 64526,
    "Union": 314164,
    "Marion": 55737,
    "Knox": 376007,
    "Pickaway": 55507,
    "Logan": 73185,
    "Madison": 49019,
    "Crawford": 50491,
    "Morrow": 144270,
    "Hardin": 30751,
    "Fayette": 159251,
    "Wyandot": 72137
}


#### Immutable Parameters #####

# County population P[i]
P = {
    "Franklin": 1356303, "Delaware": 237966, "Licking": 184898,
    "Fairfield": 167762, "Union": 71721, "Marion": 64976, "Knox": 63848,
    "Pickaway": 62158, "Logan": 46085, "Madison": 45531, "Crawford": 41626,
    "Morrow": 35927, "Hardin": 30402, "Fayette": 28782,"Wyandot": 21394
}

# Set of counties in Region 4
I = [
    "Franklin", "Delaware", "Licking", "Fairfield", "Union",
    "Marion", "Knox", "Pickaway", "Logan", "Madison",
    "Crawford", "Morrow", "Hardin", "Fayette", "Wyandot"
]

# Set of vaccination team types (administrator-to-vaccinator staffing ratios)
K = ["1to1","3to1","7to1"]

# Number of staff in each team
n_V = {"1to1": 1, "3to1": 1, "7to1": 1}
n_A = {"1to1": 1, "3to1": 3, "7to1": 7}

# Hourly vaccination rates for each team
R = {"1to1": 6, "3to1": 19, "7to1": 30}

# Work hours
hours_per_week = 40

# Maximum number of people each team can vaccinate
M = {k: R[k] * hours_per_week for k in K}

# Square footage requirement
sqft = 2500

# ----------------------------------
# Results storage
# ----------------------------------
results = []

# --------------------------
# Model
# --------------------------

m = Model("Ohio Region 4 Vaccine Planning - Week")
m.Params.OutputFlag = 0

# Decision variables
t = m.addVars(I, K, vtype=GRB.INTEGER, lb=0, name="t")
d = m.addVars(I, vtype=GRB.INTEGER, lb=0, name="d")
z = m.addVars(I, vtype=GRB.INTEGER, lb=0, name="z")
```

```
# Objective
m.setObjective(quicksum(z[i] for i in I), GRB.MAXIMIZE)

# Constraints

# 1. Vaccine supply
m.addConstr(quicksum(d[i] for i in I) <= V, name="VaccineSupply")

# 2. Vaccinator availability
m.addConstr(quicksum(n_V[k] * t[i, k] for i in I for k in K) <= E_V,
            name="TotalVaccinators")

# 3. Admin staff availability
m.addConstr(quicksum(n_A[k] * t[i, k] for i in I for k in K) <= E_A,
            name="TotalAdmin")

for i in I:

    # 4. Dose feasibility
    m.addConstr(z[i] <= d[i], name=f"FeasibleDoses[{i}]")

    # 5. Herd immunity
    m.addConstr(z[i] <= (0.8 - C[i]) * P[i], name=f"HerdImmunity[{i}]")

    # 6. Weekly participation
    m.addConstr(z[i] <= 0.1 * P[i] * (1 - C[i]), name=f"Participation[{i}]")

    # 7. Building Size
    m.addConstr(sqft * quicksum(t[i, k] for k in K) <= S[i], name=f"Building[{i}]")

    # 8. Throughput
    m.addConstr(z[i] <= quicksum(M[k] * t[i, k] for k in K), name=f"Throughput[{i}]

# 9. Fairness constraint
for i in I:
    for j in I:

        # Only compare different counties
        if i != j:

            m.addConstr((C[i] * P[i] + z[i]) * P[j] - (C[j] * P[j] + z[j]) * P[i] <
            m.addConstr((C[j] * P[j] + z[j]) * P[i] - (C[i] * P[i] + z[i]) * P[j] <

# --------------------------
#  SOlve
# --------------------------

# Solve
m.optimize()

# Store results
for i in I:
    opt_z = {i: z[i].X for i in I}
    opt_d = {i: d[i].X for i in I}
    opt_t = {(i, k): t[i, k].X for i in I for k in K}
    coverage_start = C[i]
    coverage_end   = C[i] + opt_z[i] / P[i]

    # Compute staff sent to county i
    vacc_staff_i = sum(rV[k] * opt_t[(i, k)] for k in K)
    admin_staff_i = sum(rA[k] * opt_t[(i, k)] for k in K)

    results.append({
        "county (i)": i,
        "coverage (C_i)": coverage_start,
        "updated coverage": coverage_end,
        "doses_sent (d_i)": opt_d[i],
        "vaccinated (z_i)": opt_z[i],
        "teams_1to1 (t_i,1to1)": opt_t[(i, "1to1")],
        "teams_3to1 (t_i,3to1)": opt_t[(i, "3to1")],
        "teams_7to1 (t_i,7to1)": opt_t[(i, "7to1")],
        "total vaccinators": vacc_staff_i,
        "total admin staff": admin_staff_i
    })

# ==========================================================
#                  WRITE RESULTS
# ==========================================================

with open("group22_output_1week.csv", "w", newline="") as f:
    writer = csv.DictWriter(f, fieldnames=results[0].keys())
    writer.writeheader()
    writer.writerows(results)

print("\nSaved results to group22_output_1week.csv")
```

```
Saved results to group22_output_1week.csv
```

```
In [ ]:
```