```
import itertools
import random
import math
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
from openpyxl import Workbook
from openpyxl.drawing.image import Image as XLImage
from openpyxl.styles import Border, Side, Alignment, Font, PatternFill
from google.colab import files

# ======================
# DATOS
# ======================

boxes = {
    "A": {"dims": (30,20,10), "weight":4.5, "rotatable": True, "qty": 2},
    "B": {"dims": (25,15,8),  "weight":3.2, "rotatable": False, "qty": 1},
    "C": {"dims": (20,20,12), "weight":5.0, "rotatable": True, "qty": 2},
    "D": {"dims": (18,12,6),  "weight":1.8, "rotatable": True, "qty": 3},
    "E": {"dims": (15,10,5),  "weight":1.2, "rotatable": False, "qty": 2},
    "F": {"dims": (22,18,9),  "weight":2.5, "rotatable": True, "qty": 1},
    "G": {"dims": (28,16,7),  "weight":3.9, "rotatable": True, "qty": 2},
    "H": {"dims": (12,12,12), "weight":2.2, "rotatable": True, "qty": 2},
    "I": {"dims": (35,10,8),  "weight":4.8, "rotatable": False, "qty": 1},
    "J": {"dims": (16,14,6),  "weight":1.5, "rotatable": True, "qty": 3},
}

excel_colors = {
    "A": "1F77B4","B": "FF7F0E","C": "2CA02C","D": "D62728","E": "9467BD",
    "F": "8C564B","G": "E377C2","H": "7F7F7F","I": "BCBD22","J": "17BECF"
}
```

```
plot_colors = {
    "A": "#1f77b4","B": "#ff7f0e","C": "#2ca02c","D": "#d62728","E": "#9467bd",
    "F": "#8c564b","G": "#e377c2","H": "#7f7f7f","I": "#bcbd22","J": "#17becf"
}

MAX_MASTER = (60,40,40)
MAX_WEIGHT = 55 # Increased MAX_WEIGHT to allow all boxes to be considered
PALLET = (120,100)

# ======================
# FUNCIONES
# ======================

def rotations(dim):
    return list(set(itertools.permutations(dim)))

def volume(dim):
    return dim[0]*dim[1]*dim[2]

def center_of_gravity(placements):
    total_weight = sum(p["weight"] for p in placements)
    cx = sum((p["x"] + p["dims"][0]/2)*p["weight"] for p in placements)/total_weight
    cy = sum((p["y"] + p["dims"][1]/2)*p["weight"] for p in placements)/total_weight
    cz = sum((p["z"] + p["dims"][2]/2)*p["weight"] for p in placements)/total_weight
    return (round(cx,2), round(cy,2), round(cz,2))

def pallet_efficiency(L,W):
    count_L = PALLET[0] // L
    count_W = PALLET[1] // W
    return count_L * count_W
```

```
# ===== VALIDACION DE SOPORTE =====
def has_support(new_box, placements, support_threshold=0.8):
    if new_box["z"] == 0:
        return True

    base_area = new_box["dims"][0] * new_box["dims"][1]
    support_area = 0

    for p in placements:
        if p["z"] + p["dims"][2] == new_box["z"]:
            overlap_x = max(0, min(new_box["x"]+new_box["dims"][0], p["x"]+p["dims"][0]) -
max(new_box["x"], p["x"]))
            overlap_y = max(0, min(new_box["y"]+new_box["dims"][1], p["y"]+p["dims"][1]) -
max(new_box["y"], p["y"]))
            support_area += overlap_x * overlap_y

    return (support_area / base_area) >= support_threshold

def expand_order_with_quantities():
    expanded = []
    for name, data in boxes.items():
        expanded.extend([name] * data.get("qty", 1))
    return expanded

def generate_layout(order):
    placements = []
    x = y = z = 0
    row_height = 0
    layer_height = 0

    # NUEVO: fijar una sola orientación por tipo de caja en cada solución
```

```
orientation_map = {}
for name, data in boxes.items():
    if data.get("rotatable", True):
        orientation_map[name] = random.choice(rotations(data["dims"]))
    else:
        orientation_map[name] = data["dims"]

for name in order:
    data = boxes[name]
    dims = orientation_map[name]

    if x + dims[0] > MAX_MASTER[0]:
        x = 0
        y += row_height
        row_height = 0

    if y + dims[1] > MAX_MASTER[1]:
        y = 0
        z += layer_height
        layer_height = 0

    if z + dims[2] > MAX_MASTER[2]:
        return None

    new_box = {
        "name":name,
        "dims":dims,
        "weight":data["weight"],
        "x":x,"y":y,"z":z
    }
```

```
            # VALIDAR SOPORTE ESTRUCTURAL (opcional)
            # if not has_support(new_box, placements):
            #     return None

            placements.append(new_box)

            x += dims[0]
            row_height = max(row_height, dims[1])
            layer_height = max(layer_height, dims[2])

    return placements

# ======================
# GENERAR SOLUCIONES
# ======================

solutions = []

base_order = expand_order_with_quantities()

for _ in range(3000):

    weights = [boxes[k]["weight"] for k in base_order]
    if max(weights) / min(weights) > 1.5:
        order = sorted(base_order, key=lambda k: boxes[k]["weight"], reverse=True)
    else:
        order = base_order[:]
        random.shuffle(order)

    layout = generate_layout(order)
    if layout is None:
```

```
    continue

total_volume = sum(volume(p["dims"]) for p in layout)
total_weight = sum(p["weight"] for p in layout)
if total_weight > MAX_WEIGHT:
    continue

L = max(p["x"]+p["dims"][0] for p in layout)
W = max(p["y"]+p["dims"][1] for p in layout)
H = max(p["z"]+p["dims"][2] for p in layout)

master_volume = L*W*H
efficiency = total_volume/master_volume
cg = center_of_gravity(layout)

pallet_units = pallet_efficiency(L,W)

score = (
    efficiency * 100
    + pallet_units * 2
    - cg[2] * 1.5
    - H * 0.5
)

solutions.append({
    "layout":layout,
    "L":L,"W":W,"H":H,
    "efficiency":round(efficiency*100,2),
    "cg":cg,
    "volume":master_volume,
    "weight":total_weight,
```

```
            "pallet_units":pallet_units,
            "score":score
        })

    best_solutions = {
        "MAX_EFFICIENCY": max(solutions, key=lambda s: s["efficiency"]),
        "MIN_FOOTPRINT": min(solutions, key=lambda s: s["L"]*s["W"]),
        "MIN_HEIGHT": min(solutions, key=lambda s: s["H"]),
        "LOWEST_CG": min(solutions, key=lambda s: s["cg"][2]),
        "BEST_ENGINEERING_SCORE": max(solutions, key=lambda s: s["score"])
    }

    # ======================
    # GRAFICOS
    # ======================

    def save_3d_plot(title, sol):
        fig = plt.figure(figsize=(6,6))
        ax = fig.add_subplot(111, projection='3d')

        for p in sol["layout"]:
            ax.bar3d(p["x"],p["y"],p["z"],
                     p["dims"][0],p["dims"][1],p["dims"][2],
                     color=plot_colors[p["name"]],
                     edgecolor='black',
                     alpha=0.7)

        ax.set_xlim([0, sol["L"]])
        ax.set_ylim([0, sol["W"]])
        ax.set_zlim([0, sol["H"]])
        ax.view_init(25,35)
```

```
    ax.set_title(title)

    file = f"{title}_3D.png"
    plt.savefig(file, dpi=180)
    plt.close()
    return file

def save_top_view(title, sol):
    fig, ax = plt.subplots(figsize=(6,6))
    for p in sol["layout"]:
        rect = plt.Rectangle(
            (p["x"], p["y"]),
            p["dims"][0],
            p["dims"][1],
            facecolor=plot_colors[p["name"]],
            edgecolor='black',
            alpha=0.7
        )
        ax.add_patch(rect)

        cx = p["x"] + p["dims"][0]/2
        cy = p["y"] + p["dims"][1]/2
        ax.text(cx, cy, p["name"], ha='center', va='center',
                fontsize=12, color='black', weight='bold')

    ax.set_xlim(0, sol["L"])
    ax.set_ylim(0, sol["W"])
    ax.set_title(title + " - TOP VIEW")
    ax.set_aspect('equal')

    file = f"{title}_TOP.png"
```

```
    plt.savefig(file, dpi=180)
    plt.close()
    return file

# ======================
# CREAR EXCEL
# ======================

wb = Workbook()
wb.remove(wb.active)

thin = Border(
    left=Side(style='thin'),
    right=Side(style='thin'),
    top=Side(style='thin'),
    bottom=Side(style='thin')
)

for name, sol in best_solutions.items():
    ws = wb.create_sheet(name)

    ws["A1"] = "MASTER DATA"
    ws["A1"].font = Font(bold=True, size=13)

    master_data = [
        ["Master (L x W x H)", f"{sol['L']} x {sol['W']} x {sol['H']}"],
        ["Volume", sol["volume"]],
        ["Efficiency (%)", sol["efficiency"]],
        ["Weight (kg)", sol["weight"]],
        ["Pallet Units per layer", sol["pallet_units"]],
        ["Center of Gravity (X,Y,Z)", str(sol["cg"])]
```

```
]

for row in master_data:
    ws.append(row)

for row in ws.iter_rows(min_row=2, max_row=7, min_col=1, max_col=2):
    for cell in row:
        cell.border = thin
        cell.alignment = Alignment(horizontal="center", vertical="center")

ws.append([])
ws.append(["BOX DISTRIBUTION"])
header_row = ws.max_row + 1
ws.append(["Box","L","W","H","Weight","Pos X","Pos Y","Pos Z"])

for col in range(1,9):
    cell = ws.cell(header_row, col)
    cell.border = thin
    cell.alignment = Alignment(horizontal="center", vertical="center")
    cell.font = Font(bold=True)

for p in sol["layout"]:
    ws.append([
        p["name"],
        p["dims"][0],
        p["dims"][1],
        p["dims"][2],
        p["weight"],
        p["x"],
        p["y"],
        p["z"]
```

```
        ])
        current_row = ws.max_row
        for col in range(1,9):
            cell = ws.cell(current_row, col)
            cell.border = thin
            cell.alignment = Alignment(horizontal="center", vertical="center")
            cell.fill = PatternFill(
                start_color=excel_colors[p["name"]],
                end_color=excel_colors[p["name"]],
                fill_type="solid"
            )

    img1 = XLImage(save_3d_plot(name, sol))
    img1.width = 350
    img1.height = 350
    img1.anchor = "J2"
    ws.add_image(img1)

    img2 = XLImage(save_top_view(name, sol))
    img2.width = 350
    img2.height = 350
    img2.anchor = "J25"
    ws.add_image(img2)

file_name = "packing_ENGINEERING_FULL.xlsx"
wb.save(file_name)
files.download(file_name)
```