```
import cv2
import numpy as np

# Function to handle mouse events for drawing bounding boxes
def draw_rectangle(event, x, y, flags, param):
    global drawing, top_left_pt, bottom_right_pt, bounding_boxes_too_close,
bounding_boxes_left_zone, draw_step

    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = True
        top_left_pt = (x, y)

    elif event == cv2.EVENT_LBUTTONUP:
        drawing = False
        bottom_right_pt = (x, y)
        if draw_step == 1:
            bounding_boxes_too_close.append({"coordinates": [top_left_pt,
bottom_right_pt]})
        elif draw_step == 2:
            bounding_boxes_left_zone.append({"coordinates": [top_left_pt,
bottom_right_pt]})
        cv2.rectangle(frame, top_left_pt, bottom_right_pt, (0, 255, 0), 2)
        cv2.imshow("Draw Zones", frame)

# Load YOLO
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# Load COCO class names
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f]

# Define dangerous objects
dangerous_objects = ["gun", "knife", "explosive", "scissors", "medication", "sharp
object", "alcohol", "lighter"]

# Open a connection to the webcam (use 0 for default webcam)
cap = cv2.VideoCapture(0)

# Create a window for drawing zones
cv2.namedWindow("Draw Zones")
cv2.setMouseCallback("Draw Zones", draw_rectangle)

# Initialize variables
drawing = False
top_left_pt, bottom_right_pt = (-1, -1), (-1, -1)
bounding_boxes_too_close = []
bounding_boxes_left_zone = []
draw_step = 1  # Start with drawing zones where the person should not be too close

while True:
    ret, frame = cap.read()

    if not ret:
        break

    # Draw existing bounding boxes
    if draw_step == 1:
        for bbox in bounding_boxes_too_close:
            cv2.rectangle(frame, tuple(bbox["coordinates"][0]),
tuple(bbox["coordinates"][1]), (0, 255, 0), 2)
    elif draw_step == 2:
        for bbox in bounding_boxes_left_zone:
            cv2.rectangle(frame, tuple(bbox["coordinates"][0]),
tuple(bbox["coordinates"][1]), (0, 0, 255), 2)

    cv2.imshow("Draw Zones", frame)

    key = cv2.waitKey(1)

    if key == 27:
        if draw_step == 1:
            draw_step = 2  # Move to the next step: drawing zones where the person
cannot leave
            print("Draw zones where the person cannot leave.")
        else:
            break

# Close the drawing window
cv2.destroyWindow("Draw Zones")

while True:
    ret, frame = cap.read()

    if not ret:
        break

    height, width, channels = frame.shape

    # Detecting objects using YOLO
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True,
crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)

    class_ids = []
    confidences = []
    boxes = []
    person_found = False
    dangerous_object_found = False

    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5 and classes[class_id] in dangerous_objects:
                dangerous_object_found = True

            if confidence > 0.5 and classes[class_id] == "person":
                person_found = True
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                x = int(center_x - w / 2)
                y = int(center_y - h / 2)

                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)

    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

    font = cv2.FONT_HERSHEY_PLAIN

    for i in range(len(boxes)):
        if i in indexes:
            x, y, w, h = boxes[i]
            label = f"Person {i+1}"
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(frame, label, (x, y + 30), font, 1, (0, 255, 0), 2)

            # Check if person is too close to any preset bounding box
            for bbox in bounding_boxes_too_close:
                box_coordinates = bbox["coordinates"]
                # Check if any corner of the person's bounding box is within a
certain distance from the nearest edge
                corners = [(x, y), (x + w, y), (x, y + h), (x + w, y + h)]

                for corner in corners:
                    corner_x, corner_y = corner

                    # Calculate distance from the corner to the nearest edge of the
preset bounding box
                    distance_to_edge_x = min(abs(corner_x - box_coordinates[0][0]),
abs(corner_x - box_coordinates[1][0]))
                    distance_to_edge_y = min(abs(corner_y - box_coordinates[0][1]),
abs(corner_y - box_coordinates[1][1]))

                    # Set your threshold for closeness (adjust as needed)
                    closeness_threshold = 20  # Example threshold value, modify as
needed

                    # Check if the distance is less than the threshold or if the
bounding boxes intersect
                    if (
                        distance_to_edge_x < closeness_threshold
                        or distance_to_edge_y < closeness_threshold
                        or cv2.pointPolygonTest(np.array(box_coordinates),
(corner_x, corner_y), False) >= 0
                    ):
                        print(f"Alert: {label} is too close to the drawn zone")

            # Check if person leaves any preset bounding box
            for bbox in bounding_boxes_left_zone:
                box_coordinates = bbox["coordinates"]
                if not (x > box_coordinates[0][0] and y > box_coordinates[0][1] and
x < box_coordinates[1][0] and y < box_coordinates[1][1]):
                    print(f"Alert: {label} left the drawn zone")

    if not person_found:
        print("Alert: Person not found in the scene")
    if dangerous_object_found:
        print("Alert: Dangerous object in the scene")
    cv2.imshow("Frame", frame)

    key = cv2.waitKey(1)
    if key == 27:
        break

cap.release()
cv2.destroyAllWindows()
```