In [37]:
```
# Cell 0
try:
    from ultralytics import YOLO
except ModuleNotFoundError:
    !pip install -q ultralytics
    from ultralytics import YOLO

import cv2, itertools, pathlib, numpy as np
```

In [38]:
```
# Cell 1
VIDEO   = pathlib.Path("assets/Traffic_Laramie_1.mp4")     # swap to _2 later
OUTPATH = VIDEO.with_name(VIDEO.stem + "_yolo_count.mp4")
SAVE         = True

LINE_Y       = 350        # pixel row of counting line  (tune per clip)
DIRECTION    = +1         # +1 if cars move top→bottom across the line
CAR_CLASSES  = {2,3,5,7}
CONF_THR     = 0.35
MAX_DIST     = 70         # px for matching centroids
TTL_FRAMES   = 60
CACHE_TIME = int(fps * 2)
CACHE_RADIUS = 70
```

In [39]:
```
# Cell 2
model = YOLO("yolov8n.pt")
```

In [40]:
```
# Cell 3
cap  = cv2.VideoCapture(str(VIDEO))
fps  = cap.get(cv2.CAP_PROP_FPS) or 25
W    = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
H    = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

four = cv2.VideoWriter_fourcc(*"mp4v")
vw   = cv2.VideoWriter(str(OUTPATH), four, fps, (W, H)) if SAVE else None
```

In [41]:
```
# Cell 4
nextID = itertools.count()
tracks = {}                # id → (centroid, counted?, ttl)
total  = 0
recently_counted = []
```

In [42]:
```
# Cell 5
frame_idx = 0

while True:
    ok, frame = cap.read()
    if not ok:
        break

    # -- 1. YOLO detection ---------------------------------------------------
    res = model(frame, verbose=False)[0]
    detections = []                      # [(centroid), (x,y,w,h)]
    for box, cls, conf in zip(res.boxes.xyxy.cpu().numpy(),
```

```
                            res.boxes.cls.cpu().numpy(),
                            res.boxes.conf.cpu().numpy()):
        if int(cls) in CAR_CLASSES and conf > CONF_THR:
            x1, y1, x2, y2 = box.astype(int)
            cx, cy         = (x1 + x2) // 2, (y1 + y2) // 2
            detections.append(((cx, cy), (x1, y1, x2 - x1, y2 - y1)))

    # -- 2. Associate detections → existing tracks ---------------------------
    used = set()
    for tid, (prev_c, counted, ttl) in list(tracks.items()):
        match_idx = None
        if detections:
            dists = [np.hypot(cx - prev_c[0], cy - prev_c[1])
                     for (cx, cy), _ in detections]
            match_idx = int(np.argmin(dists))
            if dists[match_idx] >= MAX_DIST:
                match_idx = None

        if match_idx is not None:                       # ------- matched -----
            (cx, cy), bbox = detections[match_idx]
            used.add(match_idx)

            # crossing test
            crossed = (not counted and
                       ((DIRECTION == +1 and prev_c[1] < LINE_Y <= cy) or
                        (DIRECTION == -1 and prev_c[1] > LINE_Y >= cy)))

            if crossed:
                # duplicate filter via recently_counted cache
                dup = any(np.hypot(cx - rx, cy - ry) < CACHE_RADIUS
                          for rx, ry, _ in recently_counted)
                if not dup:
                    total += 1
                    counted = True
                    recently_counted.append([cx, cy, CACHE_TIME])
                    print(f"COUNT frame={frame_idx:5d}  id={tid}")

            tracks[tid] = ((cx, cy), counted, TTL_FRAMES)

            # draw box & TTL
            x, y, w, h = bbox
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(frame, f"{tracks[tid][2]}", (x, y + h + 15),
                        cv2.FONT_HERSHEY_PLAIN, 1, (0, 255, 0), 1)

        else:                                           # ------- unmatched ---
            ttl -= 1
            if ttl <= 0:
                tracks.pop(tid)
            else:
                tracks[tid] = (prev_c, counted, ttl)

    # -- 3. Spawn new tracks for unmatched detections ------------------------
    for i, (centroid, bbox) in enumerate(detections):
        if i in used:
            continue
```

```
        tid = next(nextID)
        tracks[tid] = (centroid, False, TTL_FRAMES)
        x, y, w, h = bbox
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(frame, f"{TTL_FRAMES}", (x, y + h + 15),
                    cv2.FONT_HERSHEY_PLAIN, 1, (0, 255, 0), 1)

    # -- 4. Decay the recently-counted cache ---------------------------------
    recently_counted = [[x, y, t - 1] for x, y, t in recently_counted if t - 1 > 0]

    # -- 5. UI overlay & output ----------------------------------------------
    cv2.line(frame, (0, LINE_Y), (W, LINE_Y), (0, 255, 255), 2)
    cv2.putText(frame, f"Cars: {total}", (10, 40),
                cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 0, 255), 3)


    if SAVE:
        vw.write(frame)
    cv2.imshow("YOLO directional count", frame)
    if cv2.waitKey(1) & 0xFF == 27:      # ESC to quit
        break

    frame_idx += 1

cap.release()
if SAVE:
    vw.release()
cv2.destroyAllWindows()
print("Final count:", total)
print("Video saved to:", OUTPATH if SAVE else "<not saved>")
```

```
COUNT frame=  206  id=22
COUNT frame=  291  id=27
COUNT frame=  572  id=48
COUNT frame=  648  id=55
COUNT frame= 1155  id=4
COUNT frame= 2418  id=132
COUNT frame= 2445  id=134
COUNT frame= 3568  id=142
COUNT frame= 3845  id=328
COUNT frame= 3876  id=354
COUNT frame= 3893  id=360
COUNT frame= 3962  id=365
Final count: 12
Video saved to: assets\Traffic_Laramie_1_yolo_count.mp4
```