Daily Stock Report Setup (Corrected – July 2025)

Sections 1–5 as previously defined.

■ Prompt for Automated Report Generation

You are a reporting assistant. Your task is to:

1. Read an Excel file containing daily stock data. Each row includes item name, quantity, and UOM.
2. Normalize item names using a fixed alias mapping (case-insensitive).
3. Normalize UOMs (e.g. PC → pcs, CR → ctn, etc.).
4. Sum quantities per item across UOMs and apply conversion rules as needed (e.g., 1 CR = 12 bottles).
5. Fill in the report for all 28 standard items, even if quantity is 0.
6. Follow this exact output format:

DAILY STOCK REPORT - [date]

1. Mochi Durian (box) -
2. Nestum Sprinkle (pkt) -
3. Cheese Patty (pcs) -
4. Spicy Tomato Sauce (pkt) -
5. Lotus Leaf Bun (pcs) -
6. Wedges (pkt) -
7. Fries (pkt) -
8. Nugget (pcs) -
9. Butter scotch (pcs) -
10. Sesame Bun (pcs) -
11. Coleslaw (pkt) -
12. Flour (bag) -
13. Oil (ctn) -
14. Faiza Rice (kg) -
15. Sanitized Egg (pcs) -

BIB (ctn)
1. Coke Asli BIB -
2. Coke Zero BIB -
3. Sprite BIB -
4. Minute Maid BIB -
5. Heaven&Earth; BIB -
6. Fanta Straw BIB -

CSD 1.5 (bottle)
1. Coke Asli 1.5 -
2. Coke Zero 1.5 -
3. Sprite 1.5 -
4. Minute Maid -
5. Heaven & Earth 1.5 -
6. Fanta Straw 1.5 -
7. Co2 (yg ada isi) -

Fill each line with the final converted quantity after processing. Do not skip any items. If an item is not found, show 0.

Daily Stock Report Setup (with Mapping & Logic Code)

■ Alias Mapping Examples

```
'FRIES - KAI DA COATED' → 'Fries'
'C.CHEEZY WEDGES - R (NEW )' → 'Wedges'
'LOTUS LEAF BUN 320g' → 'Lotus Leaf Bun'
'COKE RASA ASLI (12X1.5L PET)' → 'Coke Asli 1.5'
'CHILLED PASTEURIZED EGG MIX' → 'Sanitized Egg'
```

■ UOM Normalization Rules

```
'CR', 'BX' → 'ctn'
'PCS', 'PC' → 'pcs'
'PK', 'PKT', 'PCK' → 'pkt'
'BT' → 'bottle'
'TK', 'TANK' → 'tank'
'KG', 'BD' → 'kg'
'GM', 'G' → 'gm'
```

■ Conversion Logic (Python-style)

```
Sanitized Egg: if UOM == 'ctn', qty * 60
Faiza Rice: if UOM == 'pkt', qty * 2.25; if gm, qty / 1000
Fries: if UOM == 'gm', qty / 2500 → pkt
Wedges: if UOM == 'gm', qty / 2268 → pkt
Flour: assume 1 bag per ctn
BIB: Coke Asli → qty / 20000ml, others → qty / 10000ml
CSD Bottles: 1 ctn = 12 bottles; BT = 1 bottle
```

■ Rounding Rules

```
UOM in ['pkt', 'pcs', 'box', 'bottle'] → round to nearest int
UOM in ['kg', 'ctn', 'bag'] → round to 2 decimal places; if .00, drop decimals
If value < 1 after conversion → display 0
```

■ Daily Report Generator - Full Python Script

```
# Daily Stock Report Generator
# Reads an Excel stock count file and outputs a structured report in your format

import pandas as pd
from collections import defaultdict

# --- CONFIGURATION ---
# Alias mapping (sample - extend as needed)
alias_map = {
    "FRIES - KAI DA COATED": "Fries",
    "C.CHEEZY WEDGES - R (NEW )": "Wedges",
    "LOTUS LEAF BUN 320g": "Lotus Leaf Bun",
    "COKE RASA ASLI (12X1.5L PET)": "Coke Asli 1.5",
    "CHILLED PASTEURIZED EGG MIX": "Sanitized Egg",
    # Add more mappings here
}

# UOM normalization
uom_map = {
    "CR": "ctn", "BX": "ctn",
    "PCS": "pcs", "PC": "pcs",
    "PK": "pkt", "PKT": "pkt", "PCK": "pkt",
    "BT": "bottle",
    "TK": "tank", "TANK": "tank",
    "KG": "kg", "BD": "kg",
    "GM": "gm", "G": "gm"
}

# Conversion rules
conversion_rules = {
    "Sanitized Egg": lambda qty, uom: qty * 60 if uom == "ctn" else qty,
    "Faiza Rice": lambda qty, uom: qty * 2.25 if uom == "pkt" else qty / 1000 if uom == "gm" else qty
    "Fries": lambda qty, uom: qty / 2500 if uom == "gm" else qty,
    "Wedges": lambda qty, uom: qty / 2268 if uom == "gm" else qty,
    "Oil": lambda qty, uom: qty / 17000 if uom == "gm" else qty,
    "Coke Asli BIB": lambda qty, uom: qty / 20000 if uom == "ml" else qty,
    "Minute Maid BIB": lambda qty, uom: qty / 10000 if uom == "ml" else qty,
    "Coke Asli 1.5": lambda qty, uom: qty * 12 if uom == "ctn" else qty,
    # Add more rules as needed
}

# Final 28 items and UOM type for formatting
report_structure = [
    ("Mochi Durian", "box"),
    ("Nestum Sprinkle", "pkt"),
    ("Cheese Patty", "pcs"),
    ("Spicy Tomato Sauce", "pkt"),
    ("Lotus Leaf Bun", "pcs"),
    ("Wedges", "pkt"),
    ("Fries", "pkt"),
    ("Nugget", "pcs"),
    ("Butter scotch", "pcs"),
    ("Sesame Bun", "pcs"),
    ("Coleslaw", "pkt"),
    ("Flour", "bag"),
    ("Oil", "ctn"),
    ("Faiza Rice", "kg"),
    ("Sanitized Egg", "pcs"),
    ("Coke Asli BIB", "ctn"),
    ("Coke Zero BIB", "ctn"),
    ("Sprite BIB", "ctn"),
    ("Minute Maid BIB", "ctn"),
    ("Heaven&Earth BIB", "ctn"),
    ("Fanta Straw BIB", "ctn"),
    ("Coke Asli 1.5", "bottle"),
    ("Coke Zero 1.5", "bottle"),
    ("Sprite 1.5", "bottle"),
    ("Minute Maid", "bottle"),
    ("Heaven & Earth 1.5", "bottle"),
    ("Fanta Straw 1.5", "bottle"),
    ("Co2 (yg ada isi)", "tank")
]
# --- PROCESSING FUNCTION ---
```

```
def generate_stock_report(excel_file):
    df = pd.read_excel(excel_file)
    df["Item Name Upper"] = df["Item Name"].str.upper()
    alias_upper = {k.upper(): v for k, v in alias_map.items()}
    df["Normalized Name"] = df["Item Name Upper"].map(alias_upper)
    df["Normalized UOM"] = df["Uom"].map(uom_map)

    # Clean rows
    df_clean = df[~df["Normalized Name"].isna() & ~df["Normalized UOM"].isna()]

    # Aggregate
    grouped = df_clean.groupby(["Normalized Name", "Normalized UOM"]).agg({"Qty": "sum"}).reset_index
    report_dict = defaultdict(float)

    for _, row in grouped.iterrows():
        name, uom, qty = row["Normalized Name"], row["Normalized UOM"], row["Qty"]
        if name in conversion_rules:
            converted = conversion_rules[name](qty, uom)
            report_dict[name] += converted if isinstance(converted, (int, float)) else 0
        else:
            report_dict[name] += qty

    # Format output
    final_report = []
    for name, uom in report_structure:
        qty = report_dict.get(name, 0)
        if uom in ["pkt", "pcs", "box", "bottle"]:
            qty = round(qty)
        elif uom in ["kg", "ctn", "bag"]:
            qty = round(qty, 2)
            if qty == int(qty):
                qty = int(qty)
        elif qty < 1:
            qty = 0
        final_report.append(f"{name} ({uom}) - {qty}")

    return final_report
```

■ Script Add-On: Copy-Paste Report Output

```
# Optional: Format output for clean copy-paste (WhatsApp, email, etc.)
def format_report_for_copy(report_list):
    output = []
    # Split into sections
    main_items = report_list[:15]
    bib_items = report_list[15:21]
    csd_items = report_list[21:]

    output.extend(main_items)
    output.append("\nBIB (ctn)")
    output.extend([f"{i+1}. {line}" for i, line in enumerate(bib_items)])
    output.append("\nCSD 1.5 (bottle)")
    output.extend([f"{i+1}. {line}" for i, line in enumerate(csd_items)])

    return "\n".join(output)

# Example usage:
# result = generate_stock_report("your_file.xlsx")
# print(format_report_for_copy(result))
```