**total_lead_time and replenishment_lead_time_to_normal didn't get populated even though the sc_name and sap_sc_code exists... -> Not seeing this issue anymore**

Some joins are not working correctly still

Missing D SKU Class -> this is for Quad Max primarily

Quick tips

Conditional formatting

Conditional formatting offers a great way to highlight important information or visually alert you when something needs attention. Learn how to use Conditional Formatting here.

Lucidchart basics

Record video

Share

Customer Zip Code to SC Mapping does not have Canadian SC

Is total_lead_time the correct column to use? calculate in weeks

## User Inputs

Order History Date Range

Country Input (US vs. CA)

Customer Zip Code to SC Mapping (user inputs CSV file)

Service Level, Annual Demand Threshold, SS Coverage Max (user inputs CSV file)

Field Location Transit Times (using total_lead_time from supply_chain.outbound_logistics.fct_obl_contracted_cost)

Replenishment Lead Time to Normal (in weeks)

Lead Time Standard Deviation for Normal (user inputs a percentage)

Lead Time Standard Deviation for Field Locations (user inputs a percentage)

## Data Ingestion

There is ONE singular order (2400071202)) which has a zip code of 33076-4819 -> this is causing all values in order history zip code to be read as a string instead of a float... following up with Kaya in oms-updates channel -> this also leads to other order's zip codes to be converted from for example 1002 to 01002 -> the zip code mapping needs to be joined as a float -> additionally not sure if the zip code to sc mapping has a mapping for Canada... bc if it did then the values wouldve been read as string not floats

Should we filter out for anything else?

Retrieve all order history within selected order date range (filter our Cancelled, Reserved, and only select MY25 and MY26) -> should we filter out Not Ready for Fulfillment

Ingest MY25 to MY26 remapping SQL query

Ingest Item Master

Ingest Product Plan

Join these 4 tables into table **merged_df (first join remapping to order history by the original config_string, then join item_master compliance and product plan compliance by the newly mapped config string; also join customer zip code mapping to get the service center a customer's zip code will be directed to)**

create a copy of merged_df as **merged_df_2, however we will group by the unique_config_string_location (this is what the final output will be built off of as we want each row to be a unique_config_string_location, and a safety stock number for each one)**

Ceate a separate notebook for Normal -> for safety stock by service center, we ignore orders out of Normal

## Safety Stock Calculator (Part A)

Calculate an Aggregated Daily Demand View (calling on merged_df, for every unique_config_string_location how many orders got placed per day) -> table is called agg_daily_demand_view_df

Perform a cross join between unique list of date range and unique_config_string_location (this is necessary so that a unique_config_string_location exists for every row order_date even if no order of that SKU was placed; with this cross join table join the agg_daily_demand_view_df to get the number of daily orders, and then calculate the Standard Deviation Daily partitioned by the unique_config_string_location

Calculate an Aggregated Weekly Demand View using the agg_daily_demand_view_df (this calculates total_num_orders_per_config_string_location_per_week, avg_orders_per_week) -> table is called agg_weekly_demand_view_df

Cross Join unique list of date range and the unique_config_string location (every order_week has a unique_config_string_location even if it did not have an order); find the standard deviation of the total_num_orders_per_config_string_location_per_week by unique_config_string_location

## Safety Stock Calculator (Part B, C, D)

Join AVG Weekly demand, Annualized Demand, and STD of Weekly Demand to merged_df_2 -> table is called **merged_df_3**

Join service_level, annual_demand_threshold, ss_coverage_max, total_lead_time to merged_df_3 -> this is called **merged_df_4**

For Lead Tim STD, I am using a percentage form for User Input -> is STD supposed to be percentage*replenishment lead time to Normal/Field? percentage * lead_time in weeks

```
def assign_replenishment_lead_time(df:pd.NA):
  df['replenishment_lead_time_to_normal'] = pd.NA

  df['lead_time_std'] = pd.NA
  df.loc[df['lead_time_std'].isin(['8101']), 'lead_time_std'] =
lead_time_std_for_normal
  df.loc[df['sap_sc_code'].isin(['8101']),
'replenishment_lead_time_to_normal'] =
replenishment_lead_time_to_normal
  df.loc[~df['sap_sc_code'].isin(['8101']),
'replenishment_lead_time_to_normal'] = df['total_lead_time']
  df.loc[~df['sap_sc_code'].isin(['8101']), 'lead_time_std'] =
lead_time_std_for_field
  return df

merged_df_5 = assign_replenishment_lead_time(merged_df_4) If the
unique_config_string_location is in Normal, it's replenishment lead time is
the user input  and its std lead time is  also a user input; If the
unique_config_string_location is in the field then call on total_lead_time
which was joined in merged_df_4 from User Inputs section; the STD is a
user input as well
```

**2026_US_R1S Tri Max_EXP-CDN_INT-SND_WHL-ODD_AUD-P01_ROOF-DGP_TON-NULL_ACTBDG-DRK_UTL-S01 -> unique_config_string_location has repeating rows**

```
import pandas as pd
from scipy.stats import norm
def calculate_safety_stock(df: pd.DataFrame):
  df['z_score'] = norm.ppf(df['service_level']).round(3)


  df['raw_safety_stock_units'] = (norm.ppf(df['service_level']).round(3) *
(df['replenishment_lead_time_to_normal'] *
(df['standard_deviation_of_weekly_demand'] ** 2)) ** 0.5 +
(df['lead_time_std'] * df['average_weekly_demand']) ** 2)


  df['rounded_safety_stock_units'] = (norm.ppf(df['service_level']).round(3)
* (df['replenishment_lead_time_to_normal'] *
(df['standard_deviation_of_weekly_demand'] ** 2)) ** 0.5 +
(df['lead_time_std'] * df['average_weekly_demand']) ** 2).round(0)


  df['rounded_safety_stock_coverage'] = (df['rounded_safety_stock_units'] /
df['average_weekly_demand']).round(1)


  return df


merged_df_6 = calculate_safety_stock(merged_df_5)


merged_df_6 is calculating the raw_safety_stock_units,
rounded_safety_stock_units, rounded_safety_stock_coverage
```

**Safety Stock Calculator**
**(D) Final SS Units**

- **1. If annualized demand <= Annual demand threshold --> Final SS Units = 0**
- **2. If raw rounded raw SS coverage >= Max SS coverage --> Max SS coverage x average weekly demand --> round result to integer value**
- **3. If raw rounded raw SS coverage < Max SS coverage --> use rounded raw SS Units**

**(D) Final SS Coverage Weeks (Hist) -> Final SS Units / Average Weekly Demand**

```
as pd
s np # import numpy for np.inf and pd.NA

x_function(row):
nnualized_demand <= annual_demand_threshold
ed_demand'] <= row['annual_demand_threshold']:


rounded_safety_stock_coverage >= ss_coverage_max
ed_safety_stock_coverage'] >= row['ss_coverage_max']:
_coverage_max'] * row['average_weekly_demand']).round(0)
rounded_safety_stock_coverage < ss_coverage_max
ed_safety_stock_coverage'] < row['ss_coverage_max']:
nded_safety_stock_units']



nal_safety_stock(df:pd.DataFrame):
opy() # Always work on a copy


stom function to create 'final_safety_stock_units' column
s the function is applied row by row
safety_stock_units'] = df_copy.apply(custom_row_function, axis=1)


al_safety_stock_coverage_weeks as a separate vectorized operation
ion by zero
safety_stock_coverage_weeks'] = (
safety_stock_units'] / df_copy['average_weekly_demand']
nf, -np.inf], np.nan).fillna(0) # Replace inf with NaN, then fill remaining NaNs with 0


: calculate_final_safety_stock(merged_df_6)
```

I'd like to do a causal experiment with this policy implementation; (ie) we keep track of which vehicles we're sending bc of our safety stock policy, and we can measure KPIs such as average lead time for customer to receive an order -> something along the lines of Interrupterd Time Series? -> to do so however, we need to be very clean and intentional with the data -> (ie) we need to be able to mark vehicles we sent as part of safety stock and which were not, and also if possible it would be nice to rollout to do a control vs treatment group type of experiment (but not sure how realistic this is). other key mertics could be % fulfilled from safety stock, how often we did no thave a configuration to fulfill a customer order

Ton of confounding variables -> (1) our demand signal data is not necessarily good -> a customer might order config A because they see it in shop, but who is to say they wouldn't have ordered config B if it had existed in shop (2) We have a lot of MY25 vehicles all over NA + there are a bunch of incentives to drive demand there, which would muddy up the picture -> maybe not able to do a real causal experiment, but we can see from a high level if certain success metrics are occuring (ie. pct of customer orders should fall, delivery lead times should also fall, etc)