hey there

Hey! What's on your mind?

I have a task i'm doing a write up on called "PFI Calc for Sales Accuracy". I';ll share my notes here:

So, although we were doing this, it was letting us pull in all of this information, because we are now moving to an ERP system, what we actually need to do is to make something that is documentation that translates what we were doing into where we're trying to go.

So before what it was was that I had essentially access to the pro forma invoice database for all the pro forma invoices. But this was one that was manually being maintained in Google Sheets. I was then taking individual Google Sheets and I was basically each of the eight salespeople within Mikano Motors, each had their own sales sheet. And then I had one that was collating all of that information to consolidating the data. And I was trying to run kind of, yeah, I was trying to run a Google Apps Script on that code, on that data. And, you know, that was it was good. I think I realized that, excuse me, using AI directly instead of using maybe like Python or Pandas or something like that. It just, it was taking a really long time to do something fairly simple. I can share the like actual code for the Google Apps Script, but essentially, the point here was that I was trying to do a number of things to gain like sales performance. How are we doing in terms of tracking for like, I was trying to look at, so what I also had was an ERP report of sales. Right. And then I wanted to see, okay, so this is the sales for this person's name. And then they themselves have this Google sheet. What's the discrepancy? How many sales have they actually done? So on and so forth. Now, hopefully to ask me any questions so that we can make like documentation of what has actually happened and where we're going. And we'll probably do some mermaid diagrams and all of this. But the point being is that from going from a situation where a salesperson asks the sales admin, generate a PFI for me, they generate it manually in this template that they also have in Google Sheets, then they log it in that Google Sheet main database. And then I take, if it's for one of those AP for it, and put it in a duplicate, which has the same heading except from the last couple, which is like status, month of sale, follow-up notes. And then I'm trying to run these things. Maybe it's because I just wasn't using Python. Maybe I should have used Python. But the point is now the team are saying that salespeople are going to be converting a quotation into a sales invoice. And that that is how we'll be able to now generate an accurate view of the conversion from a PFI to a sales invoice. And then by doing that, we'll be able to know the things like who has so many sales, what is their PFI conversion rate? How are they performing, you know, in the past 30 days? How many PFIs have they generated? How many sales have they generated? Things like what is the sales cycle? Things like what is the sales cycle? How many salespeople? And other things like that. But as you can see, because of that, I feel like I actually need to stop on this task, which was me trying to improve the accuracy of the calculations or the PFI sales. And I'll just make like documentation. So I say this is where we were. This is where we're moving to. And this is in collaboration with me and the team. How many people have they generated?

Got it — you're transitioning from a manual, spreadsheet-based system to an ERP-integrated flow, and this task ("PFI Calc for Sales Accuracy") is really about documenting that shift. Here's a distilled summary and structure for your write-up:

📌 PFI Calc for Sales Accuracy: Transition Documentation

1. Purpose

To document the evolution of PFI tracking at Mikano Motors — from Google Sheets and manual logging to a more integrated, ERP-driven system. The goal is to understand sales accuracy and performance with better alignment between PFIs issued and actual sales recorded.

---

2. Previous System (Legacy Flow)

🧾 Process Overview

- **PFI Issuance**: Salesperson asks Sales Admin to generate a Pro Forma Invoice (PFI).
- **Manual Template**: PFI created using a Google Sheets template.
- **Central Log**: Details entered manually into a master Google Sheet database.
- **Salesperson Sheets**: Each of the 8 salespeople had their own individual sheet.
- **Collation**: A consolidated sheet pulled data from each salesperson’s sheet via Google Apps Script.

⚙️ Analysis Attempt

- Compared **ERP Sales Reports** to **Google Sheet entries** to check:
  - Discrepancies between reported and logged sales.
  - Number of PFIs per salesperson vs. actual sales.
  - Follow-up performance (based on added metadata like status, notes, month).

🧠 Challenges

- Manual duplication of data
- High risk of inconsistencies
- AI/GPT took too long for simple tasks; Python/Pandas would have been faster
- Salespeople didn’t trust shared sheets (fear of lead poaching), leading to decentralized data ownership

---

3. New Direction (ERP-Integrated Flow)

✅ What’s Changing

- **Quotations** → **Sales Invoices** directly inside the ERP
- ERP now becomes the **source of truth**
- PFI generation and conversion tracking will be native within the ERP system

📊 Goals with ERP

- Track:
  - PFI → Sales Invoice conversion rates
  - Sales cycle duration
  - PFIs per salesperson in time windows (e.g. last 30 days)
  - Sales performance by person/team
- Remove the need for manual status tracking or duplicated sheets

---

4. What to Document (Next Steps)

🔧 Mermaid Diagrams

Create two flows:

- **Old Process**: Manual PFI → Sheet Log → Consolidation → Apps Script Report
- **New Process**: ERP Quotation → Conversion to Invoice → Native Reporting

📄 Documentation Sections

- "What We Were Doing"
- "Problems Identified"
- "Where We're Moving To"
- "What Success Will Look Like"
- "How to Measure Sales Accuracy in ERP"

---

5. Reflection

You were building the right thing **for the old context**, but now the goal is translation — not improvement. This task is less about refining the Google Sheets system and more about:

- Capturing the logic
- Helping the ERP team build better tracking
- Defining KPIs moving forward

Would you like me to help write the Mermaid diagrams, or do a first draft of the actual doc you can share with the team?

So this is good. I would prefer if you could break the purpose into bullet points. I would prefer if you didn't mention anything about AI GPT. You need to reword the salespeople didn't trust shared seats. You need to reword that just so it doesn't seem too accusatory. So the new direction is the conversion of quotations into sales invoices and it requires each time that they essentially go through that process every time. Instead of having quotations and then separately creating sales invoices. Where you said will be native within the ERP system. Although I understand what that means. That language is a bit too technical. Yes, the goals will allow us to do the PFI and sales invoice conversion rates. Self cycle, PFI, yeah, you're correct. Self performance by team, yep. Is there any other goals that you think would be helpful? And then you said what to document the old flow in terms of mermaid diagrams, documentation sections, what we're doing, problems identified, where we're moving to, what success. So I don't think we should have too many sections. I think we should always lean on being concise. We know that I can elaborate for hours, but that's not the point. We want to be concise. I don't know if you're able to make mermaid diagrams and maybe we don't actually need mermaid diagrams, but we can use like UTF-8 in order to make it out.

Great feedback. Here's a **tightened-up and improved version** of the write-up with all your notes applied:

📌 PFI Calc for Sales Accuracy

Purpose

- Clarify how PFIs and sales were tracked before ERP implementation
- Document the shift to the new ERP-based process
- Ensure we can still track key performance metrics during and after transition
- Align the sales team and operations on what's changing and why

---

Previous Process (Manual Flow)

🧾 Steps:

1. Salesperson requests a PFI from Sales Admin
2. PFI created using a shared Google Sheets template
3. Entry manually logged in a central PFI database (Google Sheets)
4. Each salesperson also had a personal tracking sheet
5. A consolidated sheet pulled data from all individual sheets for analysis

⚠️ Observations:

- Data was fragmented across multiple sheets
- Manual duplication led to delays and inconsistencies
- Some salespeople preferred to track independently, which reduced central visibility
- Matching PFI logs with ERP-recorded sales required manual reconciliation

---

Current Direction (ERP-Integrated Flow)

🔄 What's Changing:

- PFIs are now issued as **quotations inside the ERP system**
- Each **quotation must be converted** into a sales invoice to complete the process
- Quotations and invoices are **linked in one continuous ERP workflow**
- No need for external or duplicate data entry — everything happens in-system

📊 What We'll Now Be Able to Track:

- **PFI → Sales Invoice conversion rate** (per salesperson, per period)
- **Number of PFIs generated** in any time window
- **Number of sales closed** from those PFIs
- **Sales cycle duration** (quotation → invoice timeline)
- **Sales performance** by individual and team
- **Drop-off points** — i.e. quotations issued but not converted
- **Historical trend analysis** as more data accumulates

Summary Diagram (UTF-8 Style)

Old Process:

```pgsql
Salesperson ─→ Sales Admin ─→ Google Sheet Template
                        ↓
         Central PFI Log (Manual Entry)
                        ↓
      Consolidation Sheet + Script for Reports
                        ↓
          Manual Reconciliation with ERP Sales
```

New Process:

```sql
Salesperson ─→ ERP System ─→ Quotation
                           ↓
                    Convert to Invoice
                           ↓
                    Sales Recorded in ERP
                           ↓
               Auto-tracked Metrics & Reports
```

Core Documentation (Concise Sections Only)

1. **Legacy Flow**
2. **Why We're Changing It**

3. **New Flow (ERP-Based)**
4. **What We'll Be Able to Track Going Forward**

Let me know if you want me to output this into Notion or a Google Doc format, or if you'd like any of these sections rewritten in a more formal tone for stakeholders.

> we need a better name for this task now. "📌 PFI Calc for Sales Accuracy" is misleading

Agreed — "PFI Calc for Sales Accuracy" sounds like a tool or script, not a process documentation task.

Here are sharper, more accurate name options based on the actual purpose:

🔧 Top Task Name Suggestions

1. 📄 **Documenting the Shift: PFI Tracking to ERP Sales Flow**
2. 🔁 **PFI-to-ERP Sales Flow Transition**
3. 🛠️ **From Sheets to System: Mapping the New Sales Process**
4. 🔍 **Tracking Sales Accuracy in the ERP Era**
5. 📊 **Redefining PFI Tracking in ERP**
6. ⚙️ **Consolidating Sales Tracking: Legacy to ERP**
7. 🧭 **Transition Guide: Manual PFI Flow → ERP Integration**
8. 🗂️ **ERP Sales Flow Alignment: From PFI to Invoice**
9. ▯ **Quotation-to-Invoice Tracking Setup**
10. 📘 **Sales Documentation: PFI Conversion & Performance Metrics**

🏁 Recommended Pick

If you want something direct but clean:
"ERP Sales Flow: Transition from PFI Logging"

Or if you want something more internal/tech-minded:
"PFI-to-Invoice Workflow Transition"

Want help deciding or iterating based on the audience (ops team, devs, sales)?

> ERP Sales Flow: Transition from PFI Logging works. give me the latest writieup with the new name

Here's the updated write-up with the new title:

📘 ERP Sales Flow: Transition from PFI Logging

Purpose

- Clarify how PFIs and sales were tracked before ERP implementation
- Document the shift to the new ERP-based sales process
- Preserve key performance tracking during and after the transition
- Align all stakeholders on what's changing and why

Previous Process (Manual Flow)

🧾 Steps:

1. Salesperson requests a PFI from Sales Admin
2. Admin generates the PFI using a Google Sheets template
3. Admin logs the PFI in a central Google Sheet (PFI database)
4. Each salesperson maintained a separate personal sales sheet
5. A consolidated sheet pulled all salesperson data together
6. Google Apps Script ran analyses on the combined data

⚠️ Observations:

- Data was scattered across multiple sources
- Manual duplication slowed down reporting and led to errors
- Some salespeople preferred private tracking, reducing central visibility
- Matching logged PFIs to actual ERP sales required additional effort

---

Current Direction (ERP-Integrated Flow)

🔄 What’s Changing:

- PFIs are now issued as **quotations inside the ERP**
- Quotations must be **converted directly** into sales invoices within the same system
- This creates a **continuous, traceable sales workflow** inside the ERP
- External Google Sheet tracking is no longer needed

📊 What We’ll Now Be Able to Track:

- **PFI to Invoice conversion rates** by person and time period
- **Volume of quotations issued** by each salesperson
- **Volume of successful sales** from those quotations
- **Sales cycle duration** from quotation to invoice
- **Sales performance insights** across individuals and teams

- **Drop-off analysis**: Quotations not converted to sales
- **Historical sales trends** as data builds

Summary Diagram (UTF-8 Style)

Legacy Flow:

```pgsql
Salesperson ─→ Sales Admin ─→ Google Sheet Template
                        ↓
         Central PFI Log (Manual Entry)
                        ↓
      Consolidation Sheet + Script for Reports
                        ↓
          Manual Reconciliation with ERP Sales
```

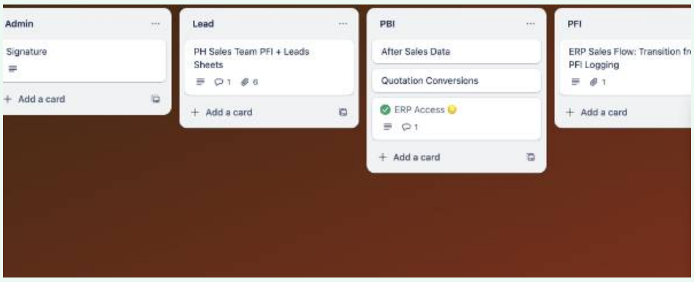
New ERP Flow:

```sql
Salesperson ─→ ERP System ─→ Quotation
                           ↓
                    Convert to Invoice
                           ↓
                    Sales Recorded in ERP
                           ↓
               Auto-tracked Metrics & Reports
```

Core Documentation Sections

1. **Legacy Flow**
2. **Why We're Changing It**
3. **New Flow (ERP-Based)**
4. **What We'll Be Able to Track**

Let me know if you'd like this exported into Notion, a Google Doc, or dropped into a shared folder for stakeholder review.



Okay, great, fantastic. So now the way that we are tracking this or the way that I've documented this is that I'm putting it in a Trello board that I have because that Trello board has like, I'll just send you a screenshot of the Trello board. So you see...

Perfect — I can see the Trello board now.

Here's a quick read on your current setup and how it connects to your documentation strategy:

✅ Your Current Trello Layout (Based on Screenshot)

Lists:

- **Admin**: Internal setup tasks (e.g., Signature)
- **Lead**: Sales team–specific sheets and lead tracking
- **PBI** (Post–Booking Insights): Focused on after-sales metrics like conversions and ERP access
- **PFI**: Where you've placed the task: **ERP Sales Flow: Transition from PFI Logging**

🧩 Recommendations for Layering the Work

Since you're already organizing your documentation across these lanes, here's a suggested way to structure the **PFI card and its relationship to other cards**:

🔷 PFI Card: ERP Sales Flow – Transition from PFI Logging

Checklist or subtasks inside this card:

- ☐ Write-up: Legacy vs. ERP Flow
- ☐ Add summary flow diagrams (UTF-style, already done)
- ☐ Link ERP metrics to current reporting goals
- ☐ Final review with ops/sales

🔷 PBI Card: Quotation Conversions

This card feels like it will *consume* the output from the PFI doc. Maybe add a checklist:

- ☐ Confirm ERP field mapping (Quotation ↔ Invoice)
- ☐ Build a view/dashboard for tracking conversion %
- ☐ Define reporting cadence (weekly/monthly)

🔄 Suggestion: Cross-link cards

To tie it all together, you could:

- Link the **PFI documentation card** inside both the **PBI** and **Lead** lists
- Use labels like ``documentation``, ``migration``, or ``metrics`` to visually connect them

- Set a due date or tag it with priority for review/finalization

Would you like a draft Trello comment or card description to paste into one of these cards?

too complex. i want to be simple and effective first

Got it. Let's strip it down.

✅ What You've Got

- A Trello board with key lists: **Admin**, **Lead**, **PBI**, **PFI**
- A single card in **PFI** titled:
  👉 **"ERP Sales Flow: Transition from PFI Logging"**
  That card = your core documentation.

🔧 Simple & Effective Setup

1. **Keep just this card** as your main doc hub
2. Inside the card, add this description (clean and minimal):

ERP Sales Flow: Transition from PFI Logging

Summary
We're moving from manually tracked PFIs in Google Sheets to a quotation-to-invoice workflow inside the ERP system.

What's Changing

- PFIs will now be ERP quotations
- Quotations must be converted to invoices
- No more Google Sheets or manual logs

Why It Matters

- We can now track PFI → Sales conversion
- Understand sales performance per person
- Reduce errors and improve visibility

Metrics to Track

- Number of PFIs (quotations) issued
- Number of invoices generated
- Conversion rate
- Sales cycle duration

- Drop-offs (quotations not invoiced)

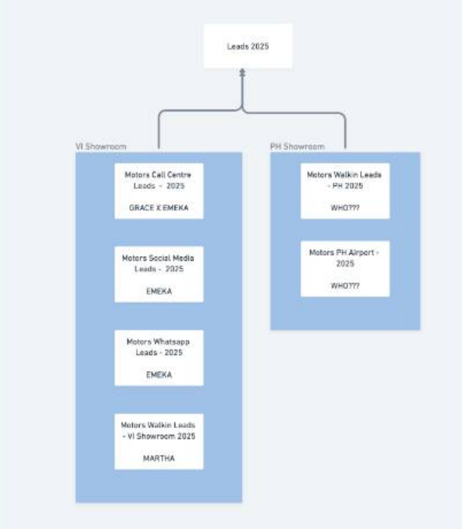
---

3. **Checklist (inside the card)**

- ☐ Confirm ERP flow is working end-to-end
- ☐ Align reporting metrics
- ☐ Share doc with sales/admin teams
- ☐ Final sign-off

That's your full setup — no cross-linking, no extra cards. Just one clear source of truth.

Want me to format this as text you can paste directly into the Trello card?


Leads 2025
VI Showroom
PH Showroom
Motors Call Centre Leads - 2025
GRACE X EMEKA
Motors Social Media Leads - 2025
EMEKA
Motors Whatsapp Leads - 2025
EMEKA
Motors Walkin Leads - VI Showroom 2025
MARTHA
Motors Walkin Leads - PH 2025
WHO???
Motors PH Airport - 2025
WHO???

So, what I actually want is for us to just be clear on what is going on essentially because now that I am no longer doing this PFI stuff, I have, there's three things, maybe I'll explain what the three things are. The first thing is having a system for logging the leads. Right, so in terms of the leads that we have, so this isn't on the ERP, this is on Google Sheet. So, we have a lead list for one, for an airport in a different state in Lake of Nigeria. We have a walk-in lead list. We have, yeah, we have a walk-in, we have a walk-in lead list. For one state, we have a walk-in lead list. For another state, we have a WhatsApp lead list. We have an Instagram lead list. We basically have a master file. Let me see if I can find the actual diagram that I made a while back now. Yeah, I'll share the screenshot with you and you can ask me questions. So, there's essentially this lead list, which is like, because you have, before a PFI is made for this person, there essentially has lead. So, it's like, how do we link lead to PFI? That's on one side. And then the other thing that I now have access to, I have access to the ability to do a sales report. All right, so I can see how many cars have we sold, like, year-to-date or in the whole time. And then I can also see the appointment register. And the appointment register is going to allow me to see the customer name, you know, the VIN number, the registration fee, all these other things. I'm even going to show you, I'll share with you a email that I wrote and sent out a couple of days ago. So, we can see how I'm tracking along that and we can start acting up on that information. I'm trying to figure out what. So, there's the leads, which I personally probably can't do today, but I just want to be clear and have the planning for that. So, one, two, there is the after-sales data slash sales data. Basically, I'll just tell you what it is. What we're trying to figure out is we've sold this many cars. We know that you're supposed to do a service roughly every six months or every 150 kilometers. So, a certain number of people should be coming and doing services every, you know, they should be coming and doing services. And they are not. And we're trying to see what that number is. And then, you know, start to do some kind of reporting on that. So, that's the second thing. And then the third thing is that I've built this inventory tool. And I'm trying to find the smartest way for us to basically transition from what they currently do, which is an Excel, which I feel like it's a bit funny. I feel like maybe that it needs to come data cleaning or something else, which, again, as soon as I start asking AI to do data cleaning, like, out the box, just Google AI Studio to make up what the data plane should be. It's not very smart and it requires, like, a lot of back and forth. So, those are the three things. So, you're going to have this note, this voice transcription that I'm sending you now. You're also going to get the email that I sent last week and a bunch of other things so we can start to plan. So, let's begin.

Email sent last week:

Good Afternoon Joelle and team,

Further to my discussion with Joelle today, please find attached two new templates and our plan for capturing PFI → After-Sales data and customer feedback. This will help us:

Track performance: measure PFI-to-invoice conversion and service-visit rates
Spot issues early: a drop-off is a customer who has not returned for service within 90 days of purchase (Mr Toni to confirm which customers)
Drive revenue: re-engage those customers before service opportunities are lost
Improve service: act on any CSAT ≤3 to fix gaps fast
1. Templates & Ownership

Tab

Data Owner

Key Fields

Sales Data Intake

Operations Team (Syam)

Customer Name, Invoice #, Invoice Date, Delivery Date, VIN, PFI #, PFI Date, Brand, Model, Trim

After-Sales Report

After-Sales Team (Mr Toni)

Customer Name, VIN, Visit #, AS Visit Date, Agent, Outcome, Issue Category

Follow-Up & Feedback

Call-Center Agents

Customer Name, VIN, Visit #, Follow-Up Date, Reason, CSAT (1–5), Recording Link, Comments

2. Key Definitions & Roles

Drop-off: no service visit within 90 days of purchase (Mr Toni to confirm)
CSAT: Customer Satisfaction score, 1 (very dissatisfied) – 5 (very satisfied)
3. Next Steps & Timelines

Template Sign-off: ASAP
Phase 1 – Manual Reporting (aim to start next Monday):
Teams populate their tabs by EOD daily.
I send a compiled summary each morning.
Phase 2 – ERP & After-Sales Coordination (ongoing):
Coordinate with Prashant and the After-Sales team to streamline service-visit data.
Call-Recording Process:
Begin with manual recordings; tool evaluation follows once agents are onboarded.
First Compiled Report (v1): Friday, 16 May
Please review the attached templates and reply with any edits or approvals ASAP so we can begin next week.

Best regards,

FT
Product & Business Intelligence Team

Thanks — everything is clear now.

You're juggling **three major post-PFI workflows**, and you're shifting into a new role where **your focus is more on systems planning and performance tracking** across lead intake, sales output, and after-sales service. Here's a clean summary to organize your thinking:

⚙️ Current Focus Areas (Post-PFI)

1. Lead Intake System (Pre-Sales)

- **Where it's tracked**: Google Sheets (not ERP)
- **What's happening**:
  - Multiple sources: Walk-ins (PH & VI), WhatsApp, Instagram, Call Center, Airport
  - Fragmented ownership — some lead lists have no clear responsible person
  - Emeka, Grace, and Martha are named; others are TBD

Goal:
Create a **unified, accountable lead logging system** that bridges into PFI issuance or quotation generation.

Next Steps:

- Identify owners for each lead list (PH Airport, PH Walk-ins still unclear)

- Map lead source → salesperson flow
- Decide whether to integrate or link this to ERP later

---

2. Sales + After-Sales Tracking

- **Where it’s tracked**:
  - ERP: Sales records
  - Manual: Service follow-ups, customer feedback

What’s happening:

- You're compiling data from three sources:
  - **Sales Intake** (from ERP, owned by Syam)
  - **After-Sales Visit Report** (owned by Mr Toni)
  - **Follow-Up Feedback** (call center)

Key Metrics:

- PFI → Sales conversion (already handled)
- **Drop-offs** = No service visit within 90 days
- **CSAT scores** = Track service quality and flag anything ≤3

Goal:
Identify service gaps and revenue leaks; implement simple, phase-based reporting to close the loop between sales and service.

---

3. Inventory System Cleanup

- **Where it’s tracked**: Excel sheets (current), being replaced with your tool
- **Problem**:
  - Excel is messy
  - Data cleaning is manual
  - Tools like AI Studio aren’t smart enough to deduce schema accurately

Goal:
Build a cleaner, reliable inventory tracker that can feed into ERP or reporting — and reduce manual overhead.

🧭 Recommended Structure Going Forward

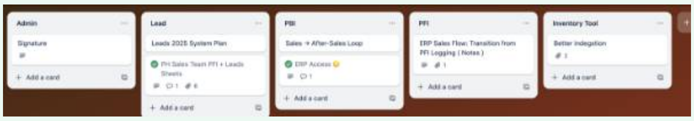
✳️ Create 3 Core Trello Cards:

1. **Leads 2025 System Plan**
   → Define sources, owners, cleanup plan

2. **Sales → After-Sales Loop**
   → Based on your email, map the drop-off + CSAT follow-up workflow
3. **Inventory Transition**
   → Decide on method: Excel cleanup vs. new system migration

---

Would you like me to write these three Trello cards out with checklist items so you can copy/paste them into your board?



+ I also have an inventory tool I'm building for ops using replit (don't mention replit)

- Where it's tracked:
- ERP: Sales records + Appointment Register
- Manual: Service follow-ups, customer feedback

Perfect. Here's a **minimal, ready-to-paste Trello-style breakdown** for your three active cards based on everything you've shared:

📌 Leads 2025 System Plan (under Lead)

Description:
Create a clear, accountable system for tracking leads across all sources before PFIs are issued.

Checklist:

- ☐ Map all current lead sources (WhatsApp, IG, Walk-ins, Airport, Call Center)
- ☐ Assign owner to each source (Grace, Emeka, Martha, etc.)
- ☐ Identify missing ownership (PH Walk-ins, PH Airport)
- ☐ Ensure consistent format across lead sheets
- ☐ Decide whether to integrate into ERP or keep in Sheets (for now)

📌 Sales → After-Sales Loop (under PBI)

Description:
Track conversion from sales to service visits, and identify customer drop-offs and satisfaction issues.

Checklist:

- ☐ Finalize 3-tab tracking templates (Sales, After-Sales, Feedback)
- ☐ Confirm definitions (e.g., drop-off = no service within 90 days)
- ☐ Get Mr Toni's feedback on identifying drop-offs

- ☐ Review CSAT scores ≤3 weekly
- ☐ Share weekly compiled report starting 16 May
- ☐ Prepare Phase 2: ERP + After-Sales team coordination

📌 Better Indigestion (under Inventory Tool)

(Consider renaming this to something clearer like “Inventory Tracker Transition”)

Description:
Improve how stock availability is tracked, moving away from messy Excel formats.

Checklist:

- ☐ Audit current Excel-based inventory files
- ☐ Identify repeated issues (e.g., manual entry errors, column mismatch)
- ☐ Build cleaner version with auto-calculated fields
- ☐ Match output format to what Ops team needs
- ☐ Test input/export with small sample set
- ☐ Prepare guidance for eventual migration

Let me know if you want this exported to a Trello card JSON format or want a shared Notion version too.

- Ensure consistent format across lead sheets -- we have a aster format already

So just to give you a bit more feedback, when it comes to this like, sales and art sales, I'm actually trying to just move forward and try to just give people stuff and then they can comment instead of be waiting for people. So because I now have, I can get a sales report, I've got the art sales appointment register for year to date, like what can I do with that already? Alright, can I make something and then present that back instead of just waiting around? Then yeah, you said to do and just just so you know the reason why I'm also saying this because this thing about review sees that scores of under three weekly. Yeah, that's great but before that I need to find a way that we're going to be getting those sees that scores right? How am I gonna contact these people? Because we don't currently have those and staff hired that can cool. So you've got to think about that. But before that what we can definitely do is like look at do the analysis of there's this many, I'm trying to decide how to do the analysis right? So like there's this many sales, there's this many visits to the aftersellers. What is the difference? I mean I have to do at a higher level of recording. What we need to see what's possible because what I'm not sure on is if I see a customer name like for example that customer, if that customer has an individual car, okay sure. But if you know if the seller happened 24 or rather in 2025 then it's not part of the sales register. If the person has a fleet or not gonna know what car it is of the fleet like you know a bank will have multiple cars. Well, that's changed.

no neeed to say I'm "not waiting" just so that you and i know whats happeneing

Crystal clear — you’re in **builder mode**, not in **“wait for feedback” mode**, and you’re right to push forward with what *can* be done now.

Here’s a refined view of your **Sales** → **After-Sales Loop** task card to reflect this approach:

📌 Sales → After-Sales Loop (under PBI)

Description:
Use existing ERP data (sales reports + appointment register) to analyze after-sales engagement and highlight missed service opportunities. Aim for insights first, feedback later.

Checklist:

- ☐ Use sales report to count total vehicles sold YTD
- ☐ Use appointment register to count total after-sales visits YTD
- ☐ Calculate % of customers that returned for service
- ☐ Flag drop-offs (customers with no visit in 90+ days post-sale)
- ☐ Identify naming inconsistencies or challenges (e.g. fleet accounts, missing VINs)
- ☐ Prepare draft analysis (summary + gaps) for team to review
- ☐ Define how CSAT will be collected (who calls, what method, what script)
- ☐ Propose a manual CSAT collection plan (temporary)
- ☐ Prep template for follow-up call logging (with placeholder CSAT)

✅ Update to Previous Lead Card Point

Replace this:

> "Ensure consistent format across lead sheets"

With this:

> ✅ **Master format already defined — confirm all sources are using it**

With this approach, you’re **pushing value forward** instead of waiting for ideal conditions — without needing to explain that explicitly to others. If you'd like, I can also help you structure the **analysis report outline** (for sales vs. service visits).

So, I'm just looking at what you said. So, this thing where you say sales report, total will be able to sold you today. And then appointment register to count total after sales visits here today. But this is like total price, where you're just saying, okay, total we have, because I think I even checked it, because this is why I'm like, oh, what do we do here? Because I can see, from the sales register, that I have, I can see, numbers are going to be exactly what the doctor and the other one are trying to say. So, when we look at some year to date sales, and if I look at quantity, the sum is 862 dollars from 517 unique deals, essentially. And then if I look at the after sales appointment register, and I look at the appointment number there are 4,605 appointments that happened so far this year. So, there are already, you can see there's way more appointments than there are, the cars that were sold. So then, I probably need to do not class or year to date, but class or total. And then I can look at year to date, is that the correct thing? I'm trying to figure out what it is before I go and add to the check that's in the seems of. And obviously this whole like, prepared draft analysis, that comes off of it first, I want to understand that. Then you said, update the previous lead card point, and sure, consistent format across the head is. Yeah, so you know that we've already done the master sheet. Let me quickly try and actually find it so you can know what the, what the layout is. Give me just a second. Okay, so, and the master sheet. Okay, so, if it can load, I'll do that on load right now. Okay, so we have column A, date, column B, customer type, that's new or existing. Column C, class of the name, column D is lead source. Lead source is hotline, Instagram, LinkedIn, Twitter, walk in, WhatsApp, airport. We have column E, phone number, column F, email address, column G, state, and the state, legos, column H, city, column I is brand. So that's the brands that this, the, the cars that we sell, like what different brands then we have model, we have trim, we also have deliverable, which is priceless or brochure. And I'm not giving you these options is that we have drop down validation and we'll achieve, we have inquiry type, which is just text, then we have status, which is sold in progress, lost, non-deal, pre-order, and then we have remarks and follow-ups. Now, if we know that essentially this is just where the leads are, but the second that it becomes like a pf by it, it's like you're not going to have the notes and stuff here. We need to think about what we're going to do. And yeah, so I see that you're talking about the leadless and you're talking about the cells after SARS-Lube, and you stop talking about the inventory tool. I don't know what you want. I can ask Reklet to give you answers to your question so you can understand what the tool currently is at and maybe that can help inform what we change in terms of the interesting like how to better ingest the data. Okay, cheers.

This was gold. Here's a sharp, structured breakdown of everything you just said — distilled into **clarified next steps**, **what you know now**, and **what's still pending**.

🧠 Updated Understanding

📈 Sales vs. After-Sales Appointments

- **Sales YTD**:
  - `862` total units sold
  - From `517` unique deals
- **After-Sales YTD**:
  - `4,605` appointments logged

👉 So: *You already have far more service visits than vehicles sold this year.*
This confirms **not all appointments = post-sale service on new cars.**

✅ What That Means

- Year-to-date **is not the right lens** to detect drop-offs.
- You need to:
  - **Filter After-Sales** to only include **vehicles sold this year**
  - Or shift to looking at **appointment count per VIN**
  - Or **track first service visit per customer post-purchase** as a proxy for success

🔧 Actionable Path Forward

📌 Sales → After-Sales Loop Card (Updated Version)

Description:
Using actual ERP data, begin exploring links between new sales and after-sales visits. Focus on patterns, not assumptions. The goal is to flag customers who bought but haven't returned — not just count everything.

Checklist (Updated):

- ☐ Sum **total sales YTD** (done: 862 units from 517 deals)
- ☐ Sum **total after-sales appointments YTD** (done: 4,605 appts)
- ☐ Attempt **VIN-level matching** between sales and visits
- ☐ Flag sales where **no visit has occurred** (possible drop-offs)
- ☐ Identify limitations (e.g., fleet names vs. individual customers)
- ☐ Create a **simple summary table** (e.g., Sold vs. First Visit)
- ☐ Draft insights doc: "How many buyers haven't come back?"
- ☐ Propose lightweight CSAT capture plan
- ☐ Skip ahead and test a **manual script/template** for call center feedback logging

---

📄 Lead Master Sheet – Clean Overview

You're already set up well. Just record this as a permanent snapshot of the current format for team reference:

Lead Sheet Columns:

- `Date`
- `Customer Type` (New / Existing)
- `Customer Name`
- `Lead Source` (Hotline, Instagram, LinkedIn, Twitter, Walk-in, WhatsApp, Airport)
- `Phone`, `Email`, `State`, `City`
- `Brand`, `Model`, `Trim`
- `Deliverable` (Price List / Brochure)
- `Inquiry Type` (Free text)
- `Status` (Sold / In Progress / Lost / Non-deal / Pre-order)
- `Remarks`, `Follow-ups`

🔁 **Current Issue**: Once a lead progresses to PFI, notes shouldn't live here anymore.
🔧 Suggestion: Create a "Status = PFI Issued" tag and start tracking deeper activity in a **separate sales tool or ERP view**.

⚙️ Inventory Tool Follow-Up

You're right — I went quiet there. Let's bring it back.

Can you ask Replit (or just describe here briefly):

- What does the tool currently accept as **input**? (CSV, Sheet, copy-paste?)
- What structure is expected?
- Where is it failing — formatting, column mismatch, missing logic?
- Are there dropdowns/validation already?

Once I have that, I'll help you spec out a **smarter ingestion pipeline** — no fluff.

Let me know if you want this grouped into a clean Notion page or just Trello card templates again.

Okay, so first of all, I'm pretty sure that he said like these options of shorter afters and only if you're able to solve this year. I do not think I'm going to be able to know that because... So let me see... let me see... I know that there is... Just a second... I know that there's a dispatch register. And on that dispatch register, I should be able to see the bins. Okay. The box... delivery... maybe it's this one, I see. Okay, so there's a delivery to job which gives me the name of the person and the date of the job, but it doesn't give me the bin number. Let me go back and see if there's something else I can do. No. Okay, so I need to figure out... Okay, so what I now have been able to get is the appointment register all the way from basically as far back as I could go. I put it to 2018, but I think the furthest is growing is 2020. And then the same for the... what's it called... dispatch register. And what this has that connects the two is the bin number. Now, one thing is that on the dispatch register, instead of it just being the bin number as a string, there's like a space and then the last three digits. But what this actually allows me to do is to have essentially a customer name. I should be able to then see the bins for that customer. All right, so that's the number of cards for each customer. And then within that thing of each card, you can see... I can see like visit number one. And I can essentially see how many visits the person has had. I don't know how to quite arrange it. But essentially, let me just do it this way. We have customer, we have bin. I'm going to make this number a diagram trying to effectively show ways. I'm trying to say we have the bins. And we have the office halls visit. And now I can effectively gain a history of a customer because I can see that this person, for any person, this is how many cards we have with that person on the system. And on top of this is how many cards we have with them. This is the details of them going to get servicing. And then we can start to see, for example, this percentage of how many people visit in X amount of time or how many people do it in Y amount of time. The big format is day, month, year for the appointment sheet. And for the dispatch register, the big format is... Okay, so what we have in this reference is that we have the delivery day and we have an invoice day. So you can see the delivery. I think what we should be using is the delivery day. And the delivery day is the day dash, like Jan did J-A-N capital J, log case A, log case N dash. And then the last two digits of the year, so 2021 is 21. And yes, essentially, I'm trying to see what the mapping is that makes sense for how I can arrange this information. And then let me just quickly look at the messages that you gave me, because of course you were saying this thing of filtering off to sell the only three vehicles sold in that year. But to see vehicles sold in that year, what I actually use is a dispatch register and look at cards we need to live in that year. So you can say, one, I can do that. Or you said shift to appointment count per VIN. Yep. Essentially appointment count per VIN and then connecting that to a customer because some customers have individuals and some customers have others. And then you said track the first service visit per customer as a post purchase as proxy for success. That doesn't work though, because as soon as someone has done their first one for one car, they would be skipped over if they bought another car. So that one doesn't work. Then you said actionable path, checklist, total sales. This is this. This is this. I think that checklist though, you're going to have to update it based on the things that I have said just now. And lead master. Yeah, you have some current issue. You progress with the PFI notes, shouldn't live there anymore, create a status PFI issue and start tracking deeper activity in a separate sales to your view. Yeah, essentially, if I can just see whether a PFI was issued and then the PFI number, then I can see the leads progressing all the way. So then I would be able to take what an export of quotations and match the quotation number to the lead number. This is my high level thinking maybe you can do better than the inventory tool. Yeah, it is quiet. Can I ask to briefly explain what does the tool currently accept? What's actually expected? Where is it failing? In formatting other dropdown for the stations already? Yeah, I can ask. I can ask and then I'll respond to you as well. Okay.

From replit (you can ask followup questions too):

Based on the code analysis, I'll now answer your questions about the import tool:

Excel/CSV Import Tool Analysis
What the tool currently accepts as input
The tool currently accepts:

Excel files (.xlsx, .xls)
CSV files (.csv)
No indication of copy-paste functionality
Expected Structure
The tool expects specific structures:

For Excel files:

Each sheet typically represents a brand
Sheets contain sections labeled as either "AVAILABLE" inventory or "TRANSIT" (pre-sale) inventory
Sections are identified by specific header keywords (like "STOCK", "SUMMARY", "MODEL" for available inventory; "PRE-SALE", "IN TRANSIT", "MODEL" for transit inventory)
Column headers are expected to follow patterns for:
Model names (identified by "MODEL" keyword)
Trim/type information (identified by "TRIM" or "TYPE" keywords)
Color information (identified by "COLOUR" or "COLOR" keywords)
Quantity data (identified by "QTY" or "QUANTITY" keywords)
For CSV files:

Expected to follow the same format as exports from the system
Contains columns for Brand, Model, Trim, Color, Quantity, Status, and Date
Failure Points
The code suggests several potential failure points:

Sheet/Brand Matching Issues:

The parser tries to match sheet names to brands, with multiple fallback strategies
If no matching brand is found for a sheet, it's skipped entirely
Column Header Detection:

The parser needs to identify the correct section headers and column mappings
It attempts to match using partial identifier matches, but might fail if headers are non-standard
Special Format Handling:

Some brands (like CHANGAN) have special handling for "dual section" sheets
Specific configuration for expected column positions
Data Format Inconsistencies:

Handling of quantity parsing (converting string values with commas)
Edge cases for special models (like GWM TANK models need special processing)
Annotations Extraction:

Text in parentheses is extracted as annotations
Some status information (like "Display" units) is derived from these annotations
Dropdowns/Validation
The front-end has limited validation:

File Type Validation:

Validates file extensions (.csv, .xlsx, .xls)
Shows error toast for invalid file types
Preview Functionality:

Files are first analyzed and previewed before actual import
Users can review changes before confirming
No Field-Level Validation:

No evidence of validation dropdowns for selecting brands, models, etc.
The system relies on matching existing data in the database
Import Process Confirmation:

Two-step process (preview then confirm)
Warning message informs users that existing data will be replaced
The import flow appears to be:

Upload file
System analyzes and previews data
User reviews changes
User confirms import
System processes changes and updates the database
The tool seems designed to work with specific Excel formats (likely from vehicle manufacturers or dealerships) and tries to be flexible in matching data, but doesn't appear to offer guided data entry options or structured templates for users to follow.

Here’s a precise, updated summary of where you are across all three systems — plus what to do next, based on your most recent input.

🔁 Sales → After-Sales Loop (Updated)

🔍 What You Have Now

- **Sales data** (via dispatch register with delivery dates)
- **Service visits** (via appointment register, 2020 onward)
- **VIN is the connector** between both datasets
- Delivery date format differs from appointment format (`` `"01-JAN-21"` `` vs `` `day/month/year` ``)

🧠 Realizations:

- **Too many visits vs. cars sold** = must **filter by delivery year**
- **Fleet customers** will create noisy data if only matched on customer name
- First-service tracking per customer doesn't work — they may own multiple vehicles

---

✅ Updated Checklist for Trello Card:

Card: *Sales → After-Sales Loop*

Checklist:

- ☑ Pull dispatch register and filter **vehicles delivered in 2025**
- ☑ Normalize VINs (e.g. strip trailing spaces, format last digits)
- ☐ Match VINs to **appointment visits**
- ☐ Create summary:
  → VINs delivered (2025 only)
  → VINs with at least one service
  → % of cars serviced within 90 days
- ☐ Group by customer to identify fleet vs individual
- ☐ Flag VINs with **no visits yet**
- ☐ Prep report draft for ops team (tables, drop-off %s, maybe a chart)
- ☐ Define path for CSAT collection (manual or script-driven)

📊 Inventory Tool

🧠 Current System:

- Accepts `` `.csv` `` and `` `.xlsx` ``
- Assumes:
    - Each sheet = one brand
    - Two sections: `` `AVAILABLE` `` and `` `TRANSIT` ``
    - Keywords must match ("MODEL", "TRIM", "QTY")
- Errors when:

    - Sheet naming doesn't match brand
    - Header labels don’t follow the format
    - Data formatting varies (e.g. commas in numbers)
    - Special model edge cases aren’t handled (like GWM TANKs)

✅ What You Should Do Next:

Card: *Inventory Tool — Better Ingestion*

Checklist:

- ☐ Identify the most **common sheet format used by ops**
- ☐ Create a **clean master template** (pre-filled with correct headers, no weird spacing)
- ☐ Build a **header-matching cheat sheet** (e.g. MODEL → MODEL NAME)
- ☐ Add fallback handling for:
    - Changan
    - Dual section models
    - Missing quantities
- ☐ Decide if annotations (e.g. “Display”) should be split into a dedicated column
- ☐ Propose: Add **basic dropdowns or field-level validation** in preview stage
- ☐ Optional: Let user **map columns** if headers are unrecognized (basic UI lift)

Would you like a mockup of the table format to use in the report? Or should I help generate a first version of the drop-off analysis sheet (e.g. VINs sold in 2025 vs. serviced)?

inventory tool is secondary. We can come back to this. Let's focus on the Sales → After-Sales Loop

Perfect — let's zero in on **Sales → After-Sales Loop**, with no distractions.

🔁 Focus Area: Sales → After-Sales Loop

✅ What You Can Already Do

Using:

- **Dispatch Register** → Get VIN, Delivery Date (filter for 2025)
- **Appointment Register** → Get VIN, Visit Date

🧩 Matching Logic:

```
plaintext

VIN (normalized) is the common key
Delivery Date (dispatch) = reference point
Visit Date (appointment) = service timeline
```

🎯 Goals:

- Identify **which cars sold in 2025** have **returned for service**
- Flag **VINs with no visit** post-delivery
- Estimate **% serviced within 90 days**
- Surface **repeat visitors**
- Group by **customer name** to expose fleets vs individuals

✍️ Trello Card (Final Text)

Card Title: Sales → After-Sales Loop

Description:
Match sales (from dispatch register) with after-sales visits (appointment register) using VIN. Focus on vehicles delivered in 2025. Identify gaps in service follow-up and begin compiling actionable insights.

✅ Checklist

- ☑ Pull dispatch register
- ☑ Filter dispatch by `` `Delivery Year = 2025` ``
- ☑ Pull appointment register (from 2020 onward)
- ☐ Normalize VIN formats (remove extra spaces, unify case)
- ☐ Match VINs from sales to appointments
- ☐ For each VIN delivered in 2025:
  - Has at least 1 visit?
  - How soon was first visit? (in days)
- ☐ Calculate:
  - Total VINs delivered in 2025
  - **with visit within 90 days**
  - **with visit after 90 days**
  - **with no visit at all**
- ☐ Group by customer:

  - **of cars per customer**
  - **of serviced cars per customer**
- ☐ Build simple output sheet:
  → VIN | Customer | Delivery Date | # of Visits | Days to First Visit
- ☐ Draft internal summary with drop-off rate
- ☐ Share as version 1 for team review

Want me to help build the matching logic or formula ideas in Sheets or Python? Or generate an example report structure?

Normalize VINs (e.g. strip trailing spaces, format last digits):

- this is for the dispatch register or appointment register
- the trailing spaces os on the appointment register
- another issue with the VIN's on Dispatch register is that when formatting around the column where the VIN's are. If someone has many cars they are all in one cell. In all instances the VIN begins with L (capital) and has a dash (-) before it. For example R9EH020921-LS4ASE2A3SA994062 or 2TZD 131115-UNTGUCN9RN333797,2TZD 131122-LJNTGUCNORN333879,2TZD 130985-LJNTGUCN1RN333440 (3 cars)

Perfect — this is the exact level of detail that unlocks clarity. Let me break down **VIN normalization rules** for both datasets and update the matching checklist accordingly.

🔧 VIN Normalization Plan

🔹 Appointment Register (Service Data)

- VINs are mostly clean but may have:
  - **Trailing spaces**
  - **Inconsistent casing**

✅ **Action:**

- `` `=TRIM(UPPER(A2))` `` (or Python equivalent)
  → Standardizes VIN for matching

🔸 Dispatch Register (Sales Data)

Two issues:

1. **Multiple VINs in one cell** for fleet deliveries

2. **VINs embedded** in a string format:
   e.g.
   `R9EH020921-LS4ASE2A3SA994062`
   `2TZD 131122-LJNTGUCNORN333879,...`

✅ **Extraction Logic:**

- VINs **start with `L`** (or other known patterns) and are **post-`-`**
- Use a regex or formula to:
  - Split by commas if multiple VINs exist
  - From each part, extract substring **after the last `-`**

Sample Excel formula:

```excel
=TRIM(RIGHT(A2,LEN(A2)-FIND("-",A2)))
```

But for multiple VINs per cell:
You'll need to **split to rows** (1 VIN per row).
In Python or Sheets:

- Split the string by `,`
- Apply extraction logic per part
- Flatten into separate rows

✅ Updated Trello Checklist Snippet:

(Under "Sales → After-Sales Loop" card)

- ☐ **Normalize Appointment Register VINs**:
  - Use `TRIM + UPPER` to clean
- ☐ **Normalize Dispatch Register VINs**:
  - Split cells with multiple VINs (by comma `,`)
  - Extract portion **after last `-`**
  - Flatten into 1 VIN per row
- ☐ Build final table:
  - `Customer | VIN | Delivery Date` (from Dispatch)
  - `VIN | Visit Date` (from Appointment)
- ☐ Use this to calculate:
  - Days from delivery to first service
  - VINs with no visit

  - Drop-off % (no visit within 90 days)

Would you like me to generate a Google Sheets formula set for this extraction and matching step-by-step? Or Python pseudocode?

- forget the trello checklist for now. let's just work
- How can i use python while still keeping the data secure and private easily? On my work computer i can't store anything. PErhaps it has to be done manually with your help and formula's instead? In which case it'll have to be excel (they don't want us using googlesheets)
- in lay terms explain =TRIM(UPPER(A2)) and =TRIM(RIGHT(A2,LEN(A2)-FIND("-",A2)))

Can you give examples of what you're saying so i understand

Got it — we'll keep this secure, local, and **Excel-only**, and I'll guide you through **manual-safe steps** without Python or cloud tools.

🧠 Step-by-Step Explanation

🟢 `=TRIM(UPPER(A2))` — *Clean a messy VIN*

This formula is used on the **appointment register**.

👇 What It Does:

1. `UPPER(A2)` → converts all letters to **uppercase**
2. `TRIM(...)` → removes **extra spaces** at the start or end

🧪 Example:

Raw VIN	Formula	Result
`ls4ase2a3sa994062`	`=TRIM(UPPER(A2))`	`LS4ASE2A3SA994062`
`LS4ASE2A3SA994062`	`=TRIM(UPPER(A2))`	`LS4ASE2A3SA994062`
`LS4ASE2A3SA994062`	`=UPPER(A2)` (optional case)	`LS4ASE2A3SA994062`

✅ Use this on your **appointment register** VIN column to clean it before matching.

🟡 `=TRIM(RIGHT(A2,LEN(A2)-FIND("-",A2)))` — *Extract single VIN from sales data*

This is for **dispatch register**, where the VIN is mixed into a longer string like:

- `R9EH020921-LS4ASE2A3SA994062`

👇 What It Does:

1. `FIND("-", A2)` → locates the **position of the dash**

2. `LEN(A2) - FIND(...)` → calculates **how many characters** come after the dash
3. `RIGHT(..., ...)` → pulls out just the text **after the dash**
4. `TRIM(...)` → removes any accidental spaces

🧪 Example:

Raw String	Formula	Result
`R9EH020921-LS4ASE2A3SA994062`	`=TRIM(RIGHT(A2,LEN(A2)-FIND("-",A2)))`	`LS4ASE2A3SA994062`
`2TZD 131122-LJNTGUCNORN333879`	`=TRIM(RIGHT(A2,LEN(A2)-FIND("-",A2)))`	`LJNTGUCNORN333879`

🚫 **Limitation**: This works only if **there is ONE VIN per cell**.

🔴 For multiple VINs in one cell (like fleets)

Example:

```
2TZD 131115-UNTGUCN9RN333797,2TZD 131122-LJNTGUCNORN333879,2TZD 130985-LJNTGUCN1RN333440
```

✅ You'll need to:

1. **Split** this into separate rows — one VIN per row
2. Then apply the **same `RIGHT()` formula** to each

How to split:
In Excel:

- Copy the cell
- Paste into Notepad
- Replace all commas with line breaks
- Paste back into Excel as multiple rows
  (or use Text to Columns > Delimited > Comma, then transpose)

Once split, apply the VIN-extraction formula to each.

🧩 Summary So Far

Data Source	Problem	Solution
Appointment Register	Trailing spaces, lowercase VINs	`=TRIM(UPPER(...))`
Dispatch Register	VIN buried in text string	`=TRIM(RIGHT(...FIND("-",A2)...))`
Dispatch Register	Multiple VINs in one cell	Split into separate rows, then apply formula

---

Would you like me to create a sample Excel template showing:

- Input

- Cleaned VINs
- Split VINs for fleets
  So you can just paste your real data into it?

- So another issue that we have with the appointment register is that I am able to generate a PDF that has the details, but I'm not able to generate the Excel file. And just for context, this PDF is 709 pages long when we are looking at from 2020, the 1st of January 2020, all the way until today, which is the 13th of May 2025.

So another thing as well, this suggestion that you have of me manually using a notepad for how many pages this would be on, what is it on again? On dispatch register. Even still, I don't like this. Like, copy, replace all commas with line breaks, paste back into, surely, surely, surely we can do this in the less manual way?

So just so we're clear, we have a couple issues. We have that the appointment details is a PDF and not an Excel file and can't get the Excel file because when it tries to pull that much as an Excel file, it just seems to crash. We have that that PDF is 709 pages long and we also have that I do not want to manually be splitting out VINs. So we want something better.

I have Professional Plus Office 2021 activated

Here's how to keep everything local in Excel—no Python, no manual Notepad hacks—and turn your 709-page PDF + messy VIN cells into clean, one-row-per-VIN tables.

---

1) Pulling the Appointment Register PDF into Excel

Office 2021 comes with **Power Query**, which can natively import tables from a PDF:

1. **Data** → **Get Data** → **From File** → **From PDF**
2. Point to your 709-page PDF.
3. In the Navigator pane, pick the table(s) that contain your appointment rows.
4. Click **Transform Data**.

Power Query will load all pages' tables into one preview. From here you can:

- **Filter** to `` `Visit Date >= 01/01/2020` ``
- **Rename** and reorder columns as needed
- **Close & Load** to dump it into a new worksheet

No more "crashing on export," and it stays completely on your machine.

---

2) Splitting & Cleaning VINs in Excel (Dispatch Register)

A) Importing the Dispatch Register

If it’s already in Excel, skip to 2B. If only PDF, repeat the above PDF-import steps.

B) Use Power Query to Split Multi-VIN Cells into Rows

1. With your dispatch sheet active, go to **Data** → **Get Data** → **From Sheet**.
2. In the Query Editor, select the **VIN** column.
3. On the ribbon: **Home** → **Split Column** → **By Delimiter**
    - Delimiter: `,` (Comma)
    - **Advanced Options**: Split into **Rows**
4. Now each VIN is on its own row.

C) Extract the Actual VIN Portion

Still in Power Query:

1. Select the split-out column.
2. **Add Column** → **Custom Column**, with formula:

```pgsql
Text.Trim( Text.AfterDelimiter([VIN], "-") )
```

3. Name it `CleanVIN`.
4. Optionally remove the original `VIN` column.
5. **Close & Load** back to Excel.

That gives you a table with one `CleanVIN` per row, ready to match against appointments.

---

3) Lay-Terms of the Key Formulas

If you ever need a quick formula instead of Power Query, here’s what they do:

3.1 Cleaning a single VIN cell

```excel
=TRIM( UPPER( A2 ) )
```

- **UPPER(A2)** → turn all letters uppercase
- **TRIM(...)** → remove extra spaces before/after

3.2 Pulling the text after the dash

```excel
=TRIM( RIGHT( A2, LEN(A2) - FIND("-", A2) ) )
```

- **FIND("-",A2)** → where the dash is
- **LEN(A2) – that** → how many chars come after it
- **RIGHT(A2,...)** → take that many chars from the right
- **TRIM(...)** → drop any stray spaces

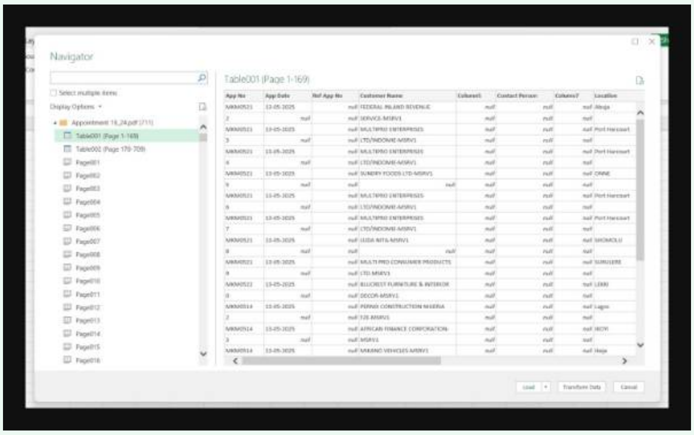
---

4) Next Steps in Excel

1. **Import PDF** (appointments) → Power Query → worksheet A
2. **Import Dispatch** → Power Query → split on `,` → extract post-dash → worksheet B
3. In a **new sheet**, write a simple `VLOOKUP` or `INDEX/MATCH` to join A & B on `CleanVIN`.
4. Calculate:
    - Days between delivery date and first visit
    - Flags for no visits (blank lookup)
    - % serviced within 90 days

All of this lives in Excel on your work PC. No cloud, no extra installs, and it scales to 700+ pages.

Let me know if you want the exact Power Query M-code snippets or a quick VBA macro for any of these steps!



After maybe 10 mins i got this screen in Power Query,

So there are some things that should be known. First of all, as I'm looking at this in Power Query, I can see that it's kind of slid out into more lines than it should. So for example, you'll have app number, which is the appointment number. They all begin with MK, or from Y and C here anyway. They begin with MK. So you can see it's MK, but then there's a second line, which is just a singular digit. And I know that this is wrong because it's essentially what it's done, and that it's continued the last digit onto a new line. Similar to how it continued the last three digits of the VIN number, which is in the column called Alter. No. So I don't know how to approach that, but you can see it in the screenshot.

Power Query often “breaks” PDF tables on odd line-wraps, but you can clean it right in the Editor—no external tools needed. Here’s a step-by-step for stitching those split rows back together:

---

1. Combine Your Page Tables Properly

When you first import via **Data** → **From PDF**, you should see “Table001 (Page 1–169)” and “Table002 (Page 170–709)”.

1. **Select only those two** (check “Select multiple items”).
2. Click **Combine** (or right-click → **Reference** on one, then append the other).
    - This yields one query with all rows, instead of dozens of “Page00x” tables.

---

2. Promote Headers & Remove Blank Rows

1. In the combined query, go to **Home** → **Use First Row as Headers**.
2. Filter out any rows that are entirely null or that contain only one digit in the **App No** column.
    - Click the little filter on **App No**, uncheck `` `(null)` `` and any single-digit entries.

---

3. Fill Down Your Appointment Numbers

Because “MK…“ got split from its trailing digit:

1. Select the **App No** column.
2. **Transform** → **Fill** → **Down**
    - Any blank cells below a valid MK code will inherit the correct value.

---

4. Clean Up Spilled-Over VIN Digits

You mentioned the VIN column (“Alter” or whatever your header is) also fractured:

1. Select that VIN column.
2. **Transform** → **Replace Values**:
    - Find: `` `^(\d)$` ``
    - Replace With: `` ` ` `` (blank)
    - *This strips out lone-digit rows.*
3. Then again **Fill Down** on that column so the full VIN migrates into any now-blank cells.

---

5. Validate & Load

1. Spot-check a few rows: each record should now have a single **App No**, a full **App Date**, and a full **VIN** on one line.
2. Click **Close & Load** to push the cleaned table into Excel.

TL;DR

1. **Combine** the two big “Table” queries.
2. **Promote headers** and **remove** any null/single-digit rows.
3. **Fill Down** on your split columns (App No, VIN).
4. **Filter out** any leftover orphan rows.
5. **Close & Load** and you have a solid, one-row-per-appointment table to match on VIN.

Give that a try and let me know if any step needs more detail—or if a tiny VBA snippet would speed up the filters/fills.

I don't see me combine, I can see I've clicked select multiple items, I've selected two tables, I can see load which I'll say drop down if I click drop down I have load and load two I have transformed data and I have cancelled. Thank you.

Okay—let’s jump into the Power Query Editor and append those two tables:

---

1. Open the Editor

1. Click **Transform Data** (not Load).
   This opens the full Power Query window.

2. Append the Two Tables

1. In the left pane, you’ll see **Table001** and **Table002**.
2. Go to the **Home** tab (ribbon at top) → click the dropdown under **Append Queries** → choose **Append Queries as New**.
3. In the dialog:
   - **Primary table**: select **Table001 (Page 1–169)**
   - **Table to append**: select **Table002 (Page 170–709)**
4. Click **OK**.
   You now have a new query (often called **Append1**) containing rows from both tables.

3. Promote & Clean

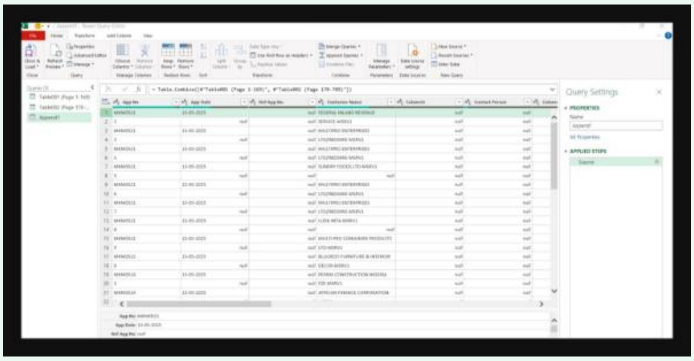
1. With **Append1** selected, go **Home** → **Use First Row as Headers**.

2. Filter out any spurious rows:
    - Click the filter arrow on **App No**, uncheck `` `(null)` `` and single digits.
3. Select the **App No** column → **Transform** → **Fill** → **Down**.
4. Do the same **Fill Down** on the VIN column.

4. Load It

1. When you see one clean row per appointment, click **Close & Load**.
2. Your full, combined, cleaned table will appear in Excel.

That appends the two PDF tables into one master table you can work with. Let me know if the **Append Queries** button is still missing or if anything in the Editor looks off!



So I think the headings are already the um... I think the headings are already the first row. What do i do now?

Great — since your headers are already correct, let's move straight into cleaning those split rows:

---

1) Filter Out the "Broken" Rows

1. **Click the filter arrow** on the **App No** column header.
2. In the filter menu, **uncheck**:
    - `` `(null)` ``
    - Any single-digit entries like `` `2` ``, `` `3` ``, `` `4` ``, etc.
3. Click **OK**.
    - This removes the "orphan" rows that only contained a stray digit.

You should now see only full `` `MKM0521` ``, `` `MKM0514` ``, etc.

---

2) Fill Down the Appointment Numbers

1. **Click the App No column** to select it.

2. On the ribbon: **Transform** → **Fill** → **Down**.
    - This copies each valid MK… code into the blank rows below it.

---

3) Repeat for the VIN Column

Identify the column with your VIN (might be called “Alter” or similar):

1. **Filter** out any rows where VIN is `` `(null)` `` or a lone digit, same as above.
2. **Select** the VIN column.
3. **Transform** → **Fill** → **Down**.

Now each row should have a full VIN next to its appointment.

---

4) Final Steps

- **Review** a handful of rows to confirm:

App No	App Date	Customer Name	VIN
MKM0521	13-05-2025	MULTIPRO ENTERPRISES	LS4ASE2A3SA994062
MKM0521	13-05-2025	MULTIPRO ENTERPRISES	UNTGUCN9RN333797

- When it looks right, hit **Close & Load** to bring this cleaned table back into Excel.

You’ll now have a proper one-row-per-appointment table—ready for matching against your cleaned Dispatch VINs. Let me know once that’s done or if you hit any snags!

So the issue is that by filtering the appointment number, sorry, the appointment date, app date, to remove null, right, because what we can see is that that null row, yes, it's the one that has the stray number. Sorry, on app number, it has, it removes the stray ones. So what I can do is I can filter. It's basically the same as me filtering app date by anything with null. What that's also going to do is remove the last three digits of the, um, alternating number, right? So I'm not sure if that's actually what we want to do.

Right—filtering out null **App Date** (or App No) is nuking the “broken” rows **and** any fragments you still need to stitch back together. Instead, let’s:

1. **Fill Down first** on all the key columns so nothing is null, then
2. **Remove rows** based on a column that really should never be blank (e.g. Customer Name), which drops only truly empty or malformed rows.

A) Fill Down All Columns That Split

In the Power Query Editor, do these in order **before** any filtering:

1. **App No** → Select column → **Transform** → **Fill** → **Down**
2. **App Date** → Select column → **Transform** → **Fill** → **Down**
3. **Customer Name** → Select → **Transform** → **Fill** → **Down**
4. **VIN** (your Alter column) → Select → **Transform** → **Fill** → **Down**

After this, **every** row—even the ones split across lines—will carry the full values from the row above.

B) Remove Truly Blank/Malformed Rows

Now that nothing is null in those columns, you can safely drop the leftover junk:

1. Click the filter arrow on **Customer Name**.
2. Uncheck `` `(null)` `` and any obviously wrong values (e.g. a single digit).
3. Click **OK**.

That removes only rows where **Customer Name** is missing—which will coincide with any rows that never had real data to begin with.

Why This Works

- **Filling down** propagates the correct App No, App Date, Customer, and VIN into split rows, so you don't lose data by filtering.
- **Filtering on Customer Name** (or another required field) drops only rows that were truly empty or garbage.

Once that's done, you'll have a clean, one-row-per-appointment table. Finish with **Close & Load**, and you're ready to match against your Dispatch VINs. Let me know how it looks!

So, I'm having a look. It says, fill down cell values to neighboring empty cells in the currently selected column. But the thing is, it's not empty. So, like, row 1 will say MKM0521, and then row 2 will have the number 2, and then row 3 will have MKM0521. And then row 4 will have 3, so it doesn't work because it's not an empty cell. (this is for App No)

Right—since those "2", "3", etc. aren't empty cells, **Fill Down** skips them. We need to first turn any stray non-"MK…" entries into true blanks, then fill.

Here's how to do that in Power Query **without writing code**:

---

1) Create a "CleanAppNo" Column via a Conditional Rule

1. In the Power Query Editor, go to **Add Column** → **Conditional Column**.
2. Name the new column `` `CleanAppNo` ``.
3. Set the rule:

   - **Column Name**: ``App No``
   - **Operator**: *begins with*
   - **Value**: ``MK``
   - **Output**: ``[App No]``
4. Add an **Else** clause: leave blank (that becomes ``null``).
5. Click **OK**.

You'll now see ``CleanAppNo`` populated only where the original starts with "MK"—all the stray digits are null.

---

2) Fill Down on ``CleanAppNo``

1. Select the **CleanAppNo** column.
2. On the ribbon: **Transform** → **Fill** → **Down**.

Now every split-row under an "MK…" will inherit the correct code.

---

3) Replace ``App No`` with ``CleanAppNo``

1. Right-click the original **App No** column → **Remove**.
2. Right-click **CleanAppNo** → **Rename** → ``App No``.

---

4) Repeat for Other Split Columns

If your VIN column also had stray fragments:

1. **Add Column** → **Conditional Column** for VIN:
   - **If** VIN **contains** ``-`` **then** VIN **else** blank
2. **Fill Down** on that new column.
3. **Remove** the old VIN column and **rename** the clean one back to "VIN."

---

5) Final Cleanup & Load

- Verify that every row has a valid ``App No`` and ``VIN``.
- Click **Home** → **Close & Load**.

Why this works:
By turning any non-"MK…" rows into true blanks, **Fill Down** now propagates your real ``App No`` into each split line. Once everything is aligned, you can drop the placeholder rows and have a perfect one-row-per-appointment table.

I'm not sure if you're understanding me. Basically, let's say it's a1 is MKM0521, and then a2 says the number 2, and then a3 says MKM0521, and a4 says 3. And what we need to do is bring a1 and a2 together, and a3 and a4 together, and a5 and a6 together. That's just for app node. And then, yeah. Does that solve by what you're saying? Because you seem to... I'm not sure if we're talking the same language.

Let's do this with a tiny Excel helper column rather than wrestling with Power Query's UI. We'll:

1. Combine the split rows in-sheet
2. Then filter down to only the "real" appointments

---

1) Bring your raw PQ output into a sheet

Assume your Power Query load dumps into **Sheet1**, columns A:D, with **App No** in **A**.

```scss
   A            B            C         D
1  App No      App Date     Customer   VIN
2  MKM0521      13-05-25    Foo Ltd    LS4…
3  2           (blank)       (blank)   (blank)
4  MKM0521      13-05-25    Foo Ltd    LS4…
5  3           (blank)       (blank)   (blank)
…
```

Rows 2-3 should be **one** record: `` `MKM05212` ``.

---

2) Helper column to stitch App No back together

In **E2**, put:

```excel
=IF(LEFT(A2,2)="MK", A2, E1 & A2)
```

- **LEFT(A2,2)="MK"**
  – checks: "Does this row look like a proper MK… code?"
- **A2** or **E1&A2**
  – if yes, take A2 fresh; if no, append the stray digit to the previous row's built code.

Then **fill (drag) E2 down** for all rows.

You'll get:

```objectivec
   A        E
2  MKM0521 MKM0521
3  2       MKM05212
4  MKM0521 MKM0521
```

```
5  3        MKM05213
…
```

---

3) Filter to only the “real” rows

Now in column **F2**, mark only the resolved rows:

```
excel

=LEFT(A2,2)="MK"
```

This returns **TRUE** on the combined rows (where A2 starts with MK). Drag down.

---

4) Copy-Paste values & filter

1. **Copy columns E:D** (your helper code plus the other fields)
2. **Paste as Values** into a new sheet.
3. **Filter** column F for **TRUE** only.
4. You now have one row per appointment, with correct `App No` in column E.

---

5) Rename & continue

- Rename **E** → **App No**
- Drop column F and original A

You’re left with a clean table: one row per appointment, correct `MKM…` codes, ready to match on VIN.

This stays entirely in Excel, is non-destructive, and avoids any “empty cell” or PQ quirks. Let me know if you’d like me to adapt it for the VIN column too!

Okay, that seems that is a solution for app number. Um, then, if I'm correct, there's this same issue. Basically, what it looks like, right, is that each row is actually related to the row below. So, instead of singular rows, they're in pairs. Right, and that's quite, that's quite an issue for everything. For everything here, like, row one and row two should be added together, right? Like, even the cluster of names split into row one and row two, and then the cluster of names split into row three and row four, which is the same thing as what happened on, um, the VIN, which fit into row one and row two. So, let me make sure that you understand what we're doing and that it works. And then, I also need to explain to me how I take this append one table and put it into Excel

1) Stitching Every Two Rows Together in Excel

You’re right—every “real” record is split across two rows for **all** columns. We can extend the same helper-column trick to grab every pair as one record:

Assume your Power Query dump is in **Sheet1** with columns A:E:

Row	A: App No	B: App Date	C: Customer Name	D: VIN	E: …
1	MKM0521	13-05-25	Foo Ltd	LS4…	…
2	2	(blank)	(blank)	(blank)	…
3	MKM0521	13-05-25	Bar Corp	XYZ…	…
4	3	(blank)	(blank)	(blank)	…
…	…	…	…	…	…

a) Helper Column for Every Field

1. In **F1**, label it **CleanAppNo**.
2. In **F2**, enter:

   ```
   excel

   =IF(LEFT(A2,2)="MK", A2, F1 & A2)
   ```

3. Drag **F2** down through all rows.
4. Repeat in **G1/H1/I1/etc.** for each field:
   - **CleanAppDate** in G2:

     ```
     excel

     =IF(LEFT($A2,2)="MK", B2, G1)
     ```

   - **CleanCustomer** in H2:

     ```
     excel

     =IF(LEFT($A2,2)="MK", C2, H1)
     ```

   - **CleanVIN** in I2:

     ```
     excel

     =IF(LEFT($A2,2)="MK", D2, I1)
     ```

   …and so on for every column you care about.

b) Mark Only the First of Each Pair

1. In **J2**, put:

   ```
   excel

   =LEFT(A2,2)="MK"
   ```

2. Drag down. Rows where **J=TRUE** are your real records.

c) Extract & Filter

1. Copy columns **F–I** (your Clean… columns).
2. **Paste as Values** into a new sheet.
3. Copy column **J** too, paste as Values next to them.
4. Filter your new sheet on **J=TRUE**.
5. Delete column J. Rename F→App No, G→App Date, etc.

You now have one row per appointment, with all fields correctly joined.

---

2) Loading "Append1" from Power Query into Excel

If you haven't already pulled **Append1** into Excel, here's how:

1. In the **Power Query Editor** (where Append1 is visible):
    - Click **Home** → **Close & Load**.
        - This auto-loads Append1 into a new worksheet.
2. **Or**, to control where it lands:
    - Click the dropdown **Close & Load** → **Close & Load To…**
    - Choose **Table** and select **Existing worksheet** or **New worksheet**.
    - Click **OK**.

That will drop your cleaned Append1 table right into Excel—after which you apply the helper-column steps above.

With these two pieces you can

1. **Bring in** all 700+ pages as one table,
2. **Stitch** each record's two rows into one,
3. **Filter** to the true appointments,
4. **Match** against your Dispatch VINs for the drop-off analysis.

Let me know if you need the exact cell-references tweaked or any of those helper formulas adjusted!

Okay, so looking at what I've now managed to do which is I've now managed to add the sheet to Excel with everything. So realistically, yes, the appointment number merging the two columns is important. The customer name merging is important. Column H is the location which is basically the first line is the actual location and the second line is blank. And then of course column L which is the altar number. First line and second line is to be merged. And that's it really. There's a bunch of other stuff but that's realistically what we need. So I'm gonna, we also have the appointment date. And there's, yeah, so what I'm gonna do, I'm gonna delete the other columns. So that's a lot clearer. To be continued. Okay, so what's the name? Yes. . . . . maybe have like a title, I'm just gonna delete them because it's just gonna become confusing. Even though some of them have things like remarks about like, you know, what the problem was, why it happened, but then what's interesting is that there is also a column that says, that's called remarks, so it's like, it's like the remarks that actually spread over, through columns. I think because right now the only thing that we're trying to do is to just match up the cells to after cells and then again, I'm gonna delete these remark ones because, as you know, it's over a long period of time and I don't know how accurate these things are, so let me just go for it and it was accurate. And then, go from there. Okay, so now what we have is, column A is the app number, appointment number, column B is the app date, it's appointment date, column C is the customer name, column D is the location, column E is the engine number, which is actually the registration pay. And column F says, column F says, number, but it's actually the pin. So I'm going to change. So I'm going to change. engine number to say, bridge number. And I'm going to say, alter number to say, VIN number. And now, essentially, what we need to do is to combine, or to append. Right? If you have row two, and row three, you append them, row four, and row five, you append them, and so on and so forth. And you're doing that, cross column A, B, C, D, E, and F. That's, that's I believe it should be. Yeah, I hear your feedback. Do I need to be helpful columns? Is there something else I can do to actually duplicate this again? Could I have raw data? I have the one where I deleted all of those columns I didn't want. Maybe I should do another one, whatever you think.

Here's a purely Excel approach to collapse every two rows into one, across all your key columns (A–F), without VBA or Power Query trickery.

---

1) Lay out a "Combined" sheet

1. Create a new worksheet, call it **CombinedAppts**.
2. In row 1, copy your headers from the cleaned sheet (App No, App Date, Customer, Location, Engine No, VIN).

---

2) Pull in every pair with INDEX formulas

In **CombinedAppts**, your first data row will be row 2. Let's say your original (split) data lives on sheet **RawAppts**, in A2:F…

CombinedAppts (cell)	Formula	What it does
A2 (App No)	**`=RawAppts!A2 & RawAppts!A3`**	Joins the "MKM…" piece in A2 with the stray digit in A3.
B2 (App Date)	**`=RawAppts!B2`**	Only the first row has the real date; second is blank.
C2 (Customer)	**`=RawAppts!C2`**	Ditto for customer name.
D2 (Location)	**`=RawAppts!D2`**	First line has it; second is blank.
E2 (Engine No)	**`=RawAppts!E2`**	First line.
F2 (VIN No)	**`=RawAppts!F2 & RawAppts!F3`**	Row 2 has start of VIN, row 3 the last digits—join them.

---

3) Copy down the formulas

- Select A2:F2
- Drag (or double-click the fill-handle) down as far as needed
- **Important**: Each CombinedAppts row n grabs RawAppts rows **2n−2** and **2n−1**.

So if you start CombinedAppts row 2 (n=2) pulling RawAppts row 2 & 3, then CombinedAppts row 3 will pull RawAppts row 4 & 5, etc.

---

4) Trim off any extra blanks

After you've filled down:

1. Select A2:F… in **CombinedAppts**
2. Copy → Paste Values in place
3. Filter column A (App No) for non-blank only
4. Delete any trailing blank rows at the bottom

📊 Example

RawAppts:

Row	A (App No)	B (Date)	…	F (VIN chunk)
2	MKM0521	13-05-2025		LS4ASE2A3SA99406
3	2	*(blank)*		2
4	MKM0522	14-05-2025		UNTGUCN9RN33379
5	3	*(blank)*		7

CombinedAppts:

Row	A (AppNo)	B (Date)	…	F (VIN)
2	MKM05212	13-05-2025		LS4ASE2A3SA994062
3	MKM05223	14-05-2025		UNTGUCN9RN333797

That gives you exactly one row per appointment, with both columns properly joined. Once you've validated a few, you can safely delete or archive **RawAppts**.

Let me know if you'd like any tweaks (e.g. formula that auto-increments the row references so you don't have to drag), or if you'd prefer a tiny VBA macro instead.

So if you start CombinedAppts row 2 (n=2) pulling RawAppts row 2 & 3, then CombinedAppts row 3 will pull RawAppts row 4 & 5, etc. - it's not doing it two and a time if that makes sense. I've done the first two row: 2+3 and 4+5 but then when i drag down i's 4+5 on row 3...

my formulas from col A to F:

```
App No =RawAppts!A2&RawAppts!A3
App Date  =RawAppts!B2&RawAppts!B3
Customer Name  =RawAppts!D2&RawAppts!D3
Location  =RawAppts!H2&RawAppts!H3
Reg No  =RawAppts!K2&RawAppts!K3
VIN. No. =RawAppts!L2&RawAppts!L3
```

Let's switch to a fully "drag-and-fill"–friendly formula using `` `INDEX` `` and `` `ROW()` `` so each CombinedAppts row automatically pulls the correct two RawAppts rows:

👉 In CombinedAppts:

1) App No (in A2)

```excel
=INDEX(RawAppts!A:A, 2*ROW()-2) & INDEX(RawAppts!A:A, 2*ROW()-1)
```

- `` `ROW()` `` returns the current row number (e.g. 2 in A2).
- `` `2*ROW()-2` `` → for row 2 gives 2, for row 3 gives 4, etc.
- `` `2*ROW()-1` `` → for row 2 gives 3, for row 3 gives 5, etc.
- `` `INDEX(..., n)` `` pulls the value from RawAppts column A at row n.

2) Appointment Date (in B2)

```excel
=INDEX(RawAppts!B:B, 2*ROW()-2)
```

(we only need the first of the pair, since the second is blank)

3) Customer Name (in C2)

```excel
=INDEX(RawAppts!D:D, 2*ROW()-2)
```

4) Location (in D2)

```excel
=INDEX(RawAppts!H:H, 2*ROW()-2)
```

5) Registration No (in E2)

```excel
=INDEX(RawAppts!K:K, 2*ROW()-2)
```

6) VIN No (in F2)

```excel
=INDEX(RawAppts!L:L, 2*ROW()-2) & INDEX(RawAppts!L:L, 2*ROW()-1)
```

🔧 How to Use

1. **Enter** those formulas in row 2 of your CombinedAppts sheet (A2:F2).
2. **Select** A2:F2 and **drag down** as far as you need.
3. **Paste as values** once done.
4. **Filter out** any rows where App No is blank (if there are extras at the bottom).

Each CombinedAppts row n will now automatically grab:

- RawAppts row 2n−2 & 2n−1 for fields that span two lines (App No, VIN)
- RawAppts row 2n−2 for single-line fields (Date, Customer, Location, Reg No)

That solves the "drag only pulls the same two rows" issue and gives you one clean record per appointment.

customer name isn't giving the full name?

If the customer name spills over into the second row, you need to concatenate both lines just like with VIN and App No. Here's the revised formula for **Customer Name** in **C2**:

```excel
=INDEX(RawAppts!D:D, 2*ROW()-2)
 & " " &
 INDEX(RawAppts!D:D, 2*ROW()-1)
```

- `` `INDEX(RawAppts!D:D, 2*ROW()-2)` `` pulls the first part of the name
- `` `INDEX(RawAppts!D:D, 2*ROW()-1)` `` pulls the continuation
- The `` `" "` `` in between ensures there's a space separating them

Once you update that in C2, drag down alongside the other columns. That will give you the full concatenated name for every appointment.

ok it's working. We need to clean the VIN column still because sometimes is more than just a VIN beginning with L and having a certain number of characters like: LS4ASE2A3SA994062

Let's weed out anything in the VIN column that isn't a "real" 17-character, "L…" VIN. We'll add a helper column and then filter.

---

1) Add a "ValidVIN" Helper Column

1. In **G1** (next to your VIN column **F**), enter header:

   ```
   nginx

   ValidVIN
   ```

2. In **G2**, put this formula:

   ```
   excel

   =AND( LEFT(TRIM(F2),1)="L",
          LEN( TRIM(F2) ) = 17 )
   ```

   - **TRIM(F2)** removes stray spaces
   - **LEFT(…,1)="L"** ensures it starts with "L"
   - **LEN(…)=17** ensures it's exactly 17 characters

3. **Fill down** G2 alongside your data.

---

2) Filter to Only Good VINs

1. Select your data (columns A–G).
2. Turn on filters (`` `Data → Filter` ``).
3. Click the filter arrow on **ValidVIN** and select only **TRUE**.

Now you'll see only rows where the VIN is exactly 17 chars long, starts with "L," and has no extra junk.

---

3) (Optional) Replace Invalid with Blank or Flag

If you prefer to keep the rows but blank out bad VINs, change the formula in G2 to:

```
excel

=IF(AND(LEFT(TRIM(F2),1)="L",LEN(TRIM(F2))=17),
    F2,
    "" )
```

Then copy–paste values back over **F** and you'll have blank VINs where data was invalid.

🧪 Example

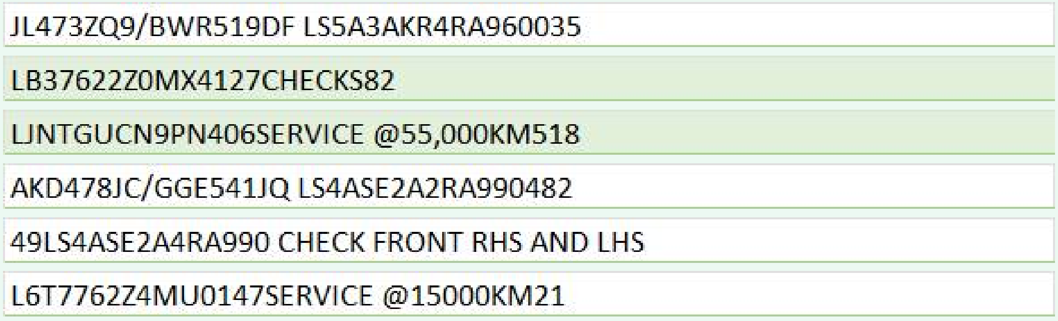
F (Raw VIN)	G (ValidVIN?)
`` `LS4ASE2A3SA994062` ``	TRUE
`` `XYZ123` ``	FALSE
`` `LJNTGUCN0RN333879` ``	TRUE
`` `LS4ASE2A3SA994062,LS4ASE…` ``	FALSE

After filtering on **G=TRUE**, you’re left only with clean VINs.

Let me know if you need help removing the helper column or collapsing your filtered results into the final sheet!

I don't want to filter the invalid ones. Because sometimes its got a valid VIN inside for example. wait for my next message

Got it—I’ll hold off. Send over an example of a mixed cell (with both valid and junk VIN text) and I’ll walk through how to pull out the good 17-character “L…” codes without losing anything else.



JL473ZQ9/BWR519DF LS5A3AKR4RA960035
LB37622Z0MX4127CHECKS82
LJNTGUCN9PN406SERVICE @55,000KM518
AKD478JC/GGE541JQ LS4ASE2A2RA990482
49LS4ASE2A4RA990 CHECK FRONT RHS AND LHS
L6T7762Z4MU0147SERVICE @15000KM21

example:

JL473ZQ9/BWR519DF LS5A3AKR4RA960035
LB37622Z0MX4127CHECKS82
LJNTGUCN9PN406SERVICE @55,000KM518
AKD478JC/GGE541JQ LS4ASE2A2RA990482
49LS4ASE2A4RA990 CHECK FRONT RHS AND LHS
L6T7762Z4MU0147SERVICE @15000KM21

as you can see our current solution doesn't work

Let’s pull those good 17-character VINs right out of the mess by moving this step back into Power Query—where it’s trivial—and then load the clean VINs back into Excel.

A) How to extract the VIN in Power Query

1. **Open** your existing query (the one that has your split-out rows).
2. **Add a Custom Column**:
    - Go to **Add Column** → **Custom Column**.
    - Name it `` `CleanVIN` ``.
    - Paste this M formula into the box:

```
m

let
  // split on common delimiters
  parts = Text.SplitAny([RawVIN], "/ @,"),
  // keep only the parts that are exactly 17 chars and start with "L"
  candidates = List.Select(parts, each Text.Length(_) = 17 and Text.StartsWith(_, "L")),
  // pick the first valid one (or null if none)
  result = if List.IsEmpty(candidates) then null else candidates{0}
in
  result
```

3. **Remove** your old VIN column (e.g. `` `[RawVIN]` ``) and **rename** `` `CleanVIN` `` → `` `VIN` ``.
4. **Close & Load** back to Excel.

Now every row will have exactly one `` `VIN` `` value—the first 17-char string beginning with "L" found in that cell, regardless of what junk surrounds it.

B) Why this works

- **Text.SplitAny** breaks the mixed string at `` `/` ``, space, `` `@` ``, comma, etc.
- **List.Select** filters to only those fragments that are 17 characters long and start with "L."
- You end up with a clean text column of valid VINs—no manual splitting, no Excel gymnastics.

Once that's in Excel, you're free to continue your `` `INDEX` ``-based joins to appointments and dispatch records. Let me know if you want the exact steps to re-run your "Combine every two rows" logic in Power Query instead of Excel, too!

- power query is closed
- will this mess up my current CombinedAppts?

You don't need to touch Power Query again, and your **CombinedAppts** sheet will remain exactly as it is. Instead, let's pull the 17-char VINs out right in your worksheet with a tiny VBA UDF—no risk to your existing data.

---

1) Add a VBA Extraction Function

1. Press **Alt + F11** to open the VBA editor.
2. **Insert** → **Module** and paste this code:

```
vba

Function ExtractVIN(txt As String) As String
    Dim re As Object, m As Object
    Set re = CreateObject("VBScript.RegExp")
    re.Pattern = "\bL[A-Z0-9]{16}\b"
    re.Global = False
    re.IgnoreCase = False

    If re.Test(txt) Then
        Set m = re.Execute(txt)
        ExtractVIN = m(0).Value
    Else
        ExtractVIN = ""  ' no valid VIN found
    End If
End Function
```

3. **Save** and close the VBA editor.

---

2) Use the Function in Excel

1. In your **CombinedAppts** sheet, insert a new column beside your raw VIN column (say column G).
2. In G2 enter:

```
excel

=ExtractVIN(F2)
```

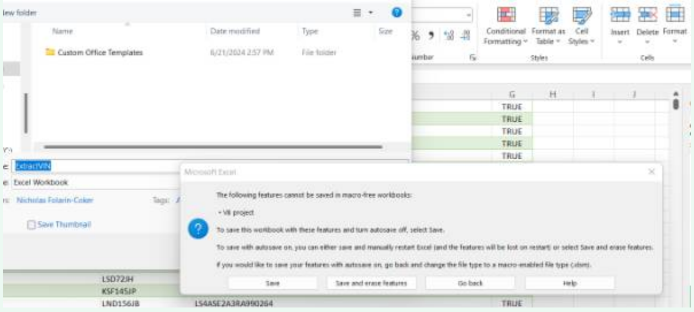
3. **Drag/fill down** G2 through all rows.
4. Column G now holds only the **first valid 17-character VIN** (starting with “L”) from whatever junk was in F2.

---

3) Swap In the Clean VINs

- Once you’ve confirmed column G is correct, you can:
    1. **Copy** G2:G…
    2. **Paste** → **Paste Values** over your original VIN column (F).
    3. **Delete** the helper column G.

Your **CombinedAppts** stays intact, and you’ve extracted just the valid VINs — no Power Query tinkering needed.



The following features cannot be saved in macro-free workbooks:

• VB project

To save this workbook with these features and turn autosave off, select Save.

To save with autosave on, you can either save and manually restart Excel (and the features will be lost on restart) or select Save and erase features.

If you would like to save your features with autosave on, go back and change the file type to a macro-enabled file type (.xlsm).

advise pls

You're seeing that because once you add VBA code, Excel insists on a **macro-enabled** file format. Here are your choices:

1. **Save as a macro-enabled workbook (.xlsm)**
    - In the Save dialog, change "Save as type" to **Excel Macro-Enabled Workbook (*.xlsm)**
    - Click **Save**
    - This preserves your VBA UDF and lets you keep Autosave on.
2. **Keep your current format but lose the macro**
    - Click **Save and erase features**
    - Your VBA code (the ExtractVIN function) will be removed, and you'll stay in a .xlsx.
    - Not recommended if you need the UDF.
3. **Disable Autosave, then Save as .xlsm**
    - If you really want to stay on OneDrive/SharePoint with Autosave, you can:
        1. Toggle **Autosave off** at the top-left.
        2. Save as .xlsm.
        3. (Optionally) toggle Autosave back on—Excel will warn that macros disable Autosave next time you open, but your macros remain.

Recommendation: Save a copy as **.xlsm**. That way you keep the ExtractVIN function intact and continue working locally without losing functionality.

ok I clicked save. whats next

Great—now that you've saved as a **.xlsm**, your `` `ExtractVIN` `` function is live. Here's what to do next:

---

1) Use the ExtractVIN Function

1. In your **CombinedAppts** sheet, insert a new column **to the right** of your existing VIN column (let's say VIN is in **F**, so insert a column **G**).
2. In **G2**, enter:

```excel
=ExtractVIN(F2)
```

3. Press **Enter**. You should see a 17-character VIN appear if one exists in F2.

---

2) Fill Down

1. Select **G2** again.
2. Double-click the small square at the bottom-right corner of the cell (the fill-handle).
    - This copies the formula down through all your rows.

---

3) Replace the Old VINs

1. Select **G2:G** (all rows of your new VIN column).
2. Press **Ctrl + C** to copy.
3. Right-click on **F2** (the first cell of your original VIN column) → **Paste Special** → **Values**.
    - This overwrites the old messy VINs with your clean, extracted ones.

---

4) Clean Up

1. Delete the helper column **G** (the one with `ExtractVIN` formulas).
2. Save your workbook.

Now your **CombinedAppts** sheet has a clean VIN column, extracted from any messy text, and you can move on to matching these VINs against your dispatch data for that drop-off analysis. Let me know if you need help with the matching step or calculating the service-return metrics!

"Formula contains unrecognized text"

It usually means Excel doesn't even see the UDF—either the VBA didn't import correctly or macros haven't been enabled. Let's verify, then fall back to a pure-formula approach.

A) Quick Macro Check

1. Press **Alt + F11** to open the VBA editor.
2. In the left pane under **VBAProject (YourWorkbook.xlsm)**, expand **Modules**.
3. You should see **Module1** (or similar). Double-click it and confirm the `ExtractVIN` code is present.
4. Close the VBA editor, then **save** and **close** the workbook.

5. Re-open it, and when prompted, click **Enable Content** (to allow macros).

Now in Excel try again:

```excel
=ExtractVIN(F2)
```

If that still gives “unrecognized text,” it means Excel isn’t picking up the VBA function.

B) Pure-Formula Fallback

We can extract the first 17-character string starting with “L” using native Excel:

1. In **G2**, enter this (adjust `F2` if your VIN is elsewhere):

   ```excel
   =LET(
     txt, TRIM(F2),
     pos, FIND("L", txt),
     vin, MID(txt, pos, 17),
     IFERROR(vin, "")
   )
   ```

2. Double-click the fill-handle on **G2** to copy it down.

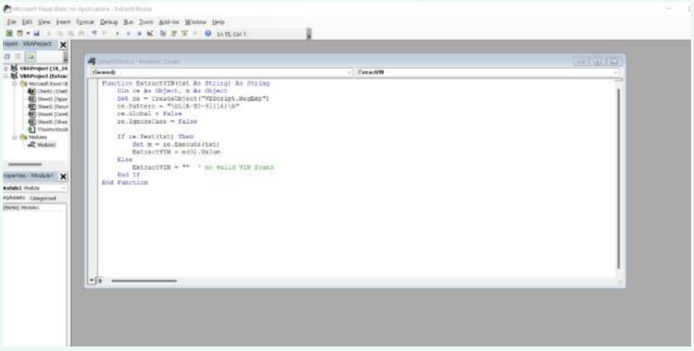
What it does

- `TRIM(F2)` cleans out extra spaces
- `FIND("L",…)` locates the first “L”
- `MID(…,pos,17)` takes that “L” plus the next 16 characters
- `IFERROR(...,"")` blanks out if there’s no “L” or fewer than 17 chars left

Once you confirm column G shows the correct VINs:

- **Copy** G2:Gn → **Paste Special** ▶️ **Values** over your old VIN column (F).
- Delete column G.

This pure-Excel approach sidesteps any VBA issues and still gets you a clean 17-char VIN out of each mixed cell. Let me know how it goes!



```
Function ExtractVIN(txt As String) As String
    Dim re As Object, m As Object
    Set re = CreateObject("VBScript.RegExp")
    re.Pattern = [illegible]
    re.Global = False
    re.IgnoreCase = False

    If re.Test(txt) Then
        Set m = re.Execute(txt)
        ExtractVIN = m(0).Value
    Else
        ExtractVIN = ""  ' no valid VIN found
    End If
End Function
```

advise

It looks like your VBA is in place, but Excel still isn't recognizing the UDF. Let's walk through the checklist to make sure macros are fully enabled and that the function lives in the right spot:

---

1) Confirm You're in a Macro-Enabled File

- You saved as `` `.xlsm` ``. Double-check the title bar says `` `YourFileName.xlsm` ``.

2) Enable Macros on Open

1. Close the workbook entirely.
2. Re-open it.
3. If you see a **yellow security warning** ("Macros have been disabled"), click **Enable Content** (or **Enable Macros**).

   Tip: If no warning appears, go to **File** → **Options** → **Trust Center** → **Trust Center Settings** → **Macro Settings**, and temporarily select **"Enable all macros"** and **"Trust access to the VBA project object model"**. Click OK, then restart Excel and re-open your `` `.xlsm` ``.

3) Verify the Function Is in a Standard Module

1. Press **Alt+F11** to open the VBA editor.
2. In the Project Explorer on the left, expand **VBAProject (YourWorkbook.xlsm)**.
3. Under **Modules**, you should see **Module1** (or whatever you named it).
4. Double-click **Module1** and confirm the `` `Function ExtractVIN(txt As String) As String` `` code is **exactly** there—no typos.

4) Call It from a Cell

- Back in Excel, in G2 type:

  ```
  excel

  =ExtractVIN(F2)
  ```

- **Do not** prefix with anything (no `` `Module1.` `` or workbook name). Just `` `=ExtractVIN(F2)` ``.

If it still shows “unrecognized text,” try:

- **Insert** a brand-new module:
    1. In VBA editor: **Insert** → **Module**
    2. Paste the function there
    3. Save, close, re-open, **Enable Macros**

Fallback: Pure-Excel LET Formula

If after all that Excel still won’t see the UDF, use this in G2 instead:

```excel
=LET(
  txt, TRIM(F2),
  pos, FIND("L", txt),
  vin, MID(txt, pos, 17),
  IFERROR(vin, "")
)
```

That will extract your “L…” + 16 chars wherever it appears. Copy down, paste-values over F, and delete G.

Give those steps a try—especially the **Trust Center/Macro Settings** and **re-inserting the module**. Let me know where you land!

Explain more: • You saved as .xlsm. Double-check the title bar says YourFileName.xlsm.

When you save a workbook with VBA code, you must use the **Excel Macro-Enabled Workbook** format so Excel knows to preserve and allow your macros. Here’s how to confirm and, if needed, fix it:

---

1. Check the Extension in the Title Bar

1. Look at the very top of your Excel window—where it shows the file name.
2. You should see something like:

    ```
    ExtractVIN.xlsm – Excel
    ```

    If it instead says `` `.xlsx` ``, `` `.xls` `` or just the file name without “.xlsm,” then Excel is not in the macro-enabled format.

---

2. Save As Macro-Enabled (if it isn’t already)

1. Click **File** → **Save As**.
2. Pick your desired folder.

3. In the **Save as type** dropdown, choose **Excel Macro-Enabled Workbook (*.xlsm)**.
4. Rename the file if you like, but keep the **.xlsm** extension.
5. Click **Save**.

Now your title bar should update to show `` `YourFileName.xlsm` ``.

---

3. Verify via File Properties

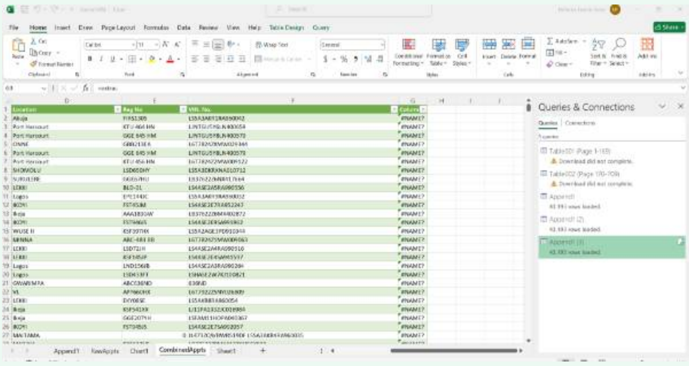
1. In Excel, click **File** → **Info**.
2. On the right, you'll see **"Properties"** at the top; click the dropdown arrow and select **Advanced Properties**.
3. In the dialog, under the **General** tab, look at **Type of file:**

   ```
   mathematica

   Microsoft Excel Macro-Enabled Workbook
   ```

   That confirms you're in the right format.

Once you see **.xlsm** in the title bar (or confirm via File → Info), your VBA UDFs will be recognized. If you don't see it yet, repeat the Save As step carefully.



wrong?

Because Excel still doesn't "see" the VBA function, let's switch over to a pure-formula extraction—no macros needed. Paste this into **G2** (assuming your dirty VIN is in **F2**) and fill down:

```
excel

=LET(
  txt, TRIM(F2),
  pos, FIND("L", txt),
  vin, MID(txt, pos, 17),
  IF(LEN(vin)=17, vin, "")
)
```

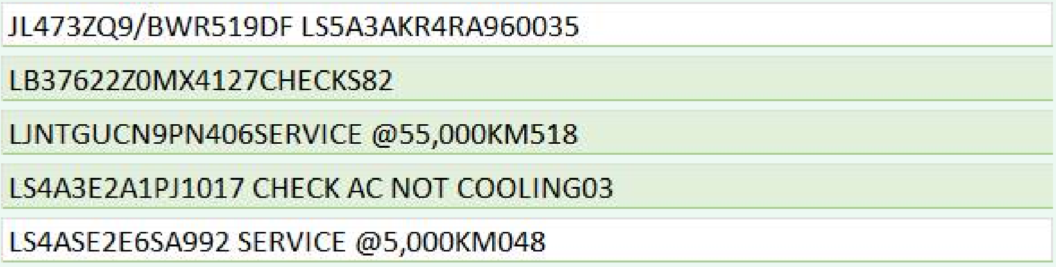
How it works

1. **TRIM(F2)** cleans leading/trailing spaces.
2. **FIND("L",txt)** locates the first “L” in the text.
3. **MID(txt,pos,17)** takes that “L” plus the next 16 characters.
4. **IF(LEN(vin)=17,vin,"")** ensures we only keep real 17-character VINs; everything else becomes blank.

Steps

1. **Enter** that formula in G2.
2. **Double-click** the fill-handle to copy it down.
3. **Copy** G2:Gn → **Paste Special ▸ Values** over your original VIN column (F).
4. **Delete** column G.

You’ll end up with a clean 17-character VIN in every row—no VBA or Power Query required.



JL473ZQ9/BWR519DF LS5A3AKR4RA960035
LB37622Z0MX4127CHECKS82
LJNTGUCN9PN406SERVICE @55,000KM518
LS4A3E2A1PJ1017 CHECK AC NOT COOLING03
LS4ASE2E6SA992 SERVICE @5,000KM048

better but didn't work for:

JL473ZQ9/BWR519DF LS5A3AKR4RA960035
LB37622Z0MX4127CHECKS82
LJNTGUCN9PN406SERVICE @55,000KM518
LS4A3E2A1PJ1017 CHECK AC NOT COOLING03
LS4ASE2E6SA992 SERVICE @5,000KM048

At this point, bending Excel formulas to catch every edge case is going to get deeply fragile—especially when your “L…” codes can sit anywhere in the string, mixed with text like “SERVICE @5,000KM048.”

The most bullet-proof way is to hand this off to Power Query one more time (it’s still in your workbook), extract exactly the 17-char VINs with a tiny M-script, then drop the cleaned table back into Excel. You won’t lose your CombinedAppts sheet—you’ll simply refresh it against a new, clean source.

---

1) Re-open Power Query on your Append1 data

1. Go to **Data** → **Queries & Connections** on the ribbon.

2. In the side pane, right-click **Append1 (3)** → **Edit**.
   This opens the Power Query Editor on the combined appointment table.

---

2) Add a “CleanVIN” column with a regex extract

1. In the Editor, click **Add Column** → **Custom Column**.
2. Name it **CleanVIN** and paste exactly this M-formula:

```m
let
  // take your raw appended VIN column (rename if needed)
  raw = [VIN],
  // split out by any non-alphanumeric to isolate candidate tokens
  tokens = List.Select(
    Text.SplitAny(raw, " /,@"),
    each Text.Length(_) = 17 and Text.StartsWith(_, "L")
  ),
  // first valid token or null
  result = if List.IsEmpty(tokens) then null else tokens{0}
in
  result
```

3. Click **OK**.
4. Remove the old **VIN** column: right-click it → **Remove**.
5. Rename **CleanVIN** → **VIN** by right-clicking the header.

---

3) Load the cleaned table back into Excel

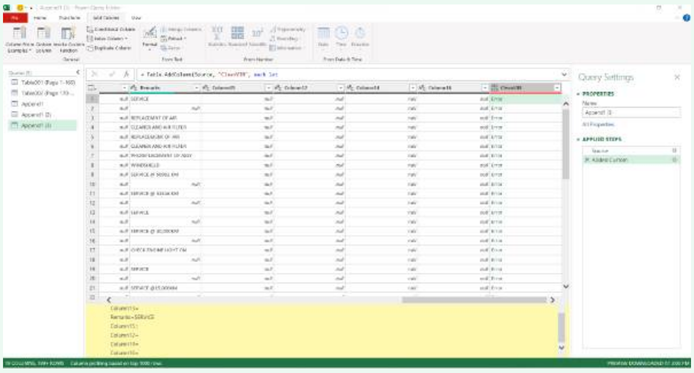
1. Click **Home** → **Close & Load To…**
2. Choose **Existing Worksheet** and pick a blank spot (or a new sheet).
3. Click **OK**.

Now you have a fresh table where **VIN** is guaranteed to be the 17-char code (the first “L…” token found). Your **CombinedAppts** sheet is untouched—you'll just use this new sheet for your matching and analytics, or you can copy/paste its values back over.

Why this works

- **Power Query** can cleanly parse and filter out exactly the 17-character token you want, no matter what else is in the cell.
- It's entirely local—no cloud, no external scripts.
- You avoid wrestling with 100+ different Excel-formula edge cases.

Give that a try and let me know how the new **VIN** column looks when you refresh!



```
Expression.Error: The field 'VIN' of the record wasn't found.
Details:
    App No=MKM0521
    App Date=13-05-2025
    Ref App No=
    Customer Name=FEDERAL INLAND REVENUE
    Column5=
    Contact Person=
    Column7=
    Location=Abuja
    Amount=
    Capacity=1.4
    Engine No=FIRS1305
    Alter. No.=LS5A3AKR1RA960
    Column13=
    Remarks=SERVICE
    Column15=
    Column12=
    Column14=
    Column16=
```

The error means Power Query can't find a column literally called `VIN` in your table — yours is still named **Alter. No.** (or whatever shows in your column header).

You have two simple fixes:

A) Rename the "Alter. No." column to VIN

1. In the Power Query Editor, right-click the **Alter. No.** header → **Rename** → type `VIN` → **Enter**.
2. Now re-run your Custom Column step (or remove & re-add it). Since the column is now `VIN`, the M-script will find it.

B) Adjust the Custom-Column code to point at "Alter. No."

If you'd rather keep the original header name, edit your formula:

1. Click the gear icon next to **Added Custom** in the APPLIED STEPS pane.

2. In the formula box, replace the line

   ```
   m

   raw = [VIN],
   ```

   with

   ```
   m

   raw = [#"Alter. No."],
   ```

   (including the `#` and quotes if the name contains spaces or punctuation).
3. Click **OK** and you'll see `CleanVIN` populate correctly.

Once you do one of those, the Extract-first-17-chars logic will run without the "field not found" error. Then just **Close & Load** and use the new clean VIN column in Excel.

[Expression.Error] We cannot convert the value null to type Text.

what colum are you thinking VIN is in?

The error is coming from trying to run `Text.SplitAny` on a `null` (empty) cell. We need to guard against that before doing any text operations. Also, your VIN column in the navigator is called **Alter. No.**, not `VIN`. Let's fix both:

---

1) Reference the correct column name

In your Custom Column step, use:

```
m

raw = [#"Alter. No."],
```

instead of `[VIN]`.

---

2) Guard against nulls

Wrap that in an `if` so you only run the split on actual text:

```
m

= Table.AddColumn(
    Source,
    "CleanVIN",
    each
      let
```

```
        raw = if [#"Alter. No."] = null then "" else [#"Alter. No."],
        tokens = List.Select(
          Text.SplitAny(raw, "/ ,@"),
          each Text.Length(_) = 17 and Text.StartsWith(_, "L")
        ),
        result = if List.IsEmpty(tokens) then null else tokens{0}
      in
        result
)
```

What this does

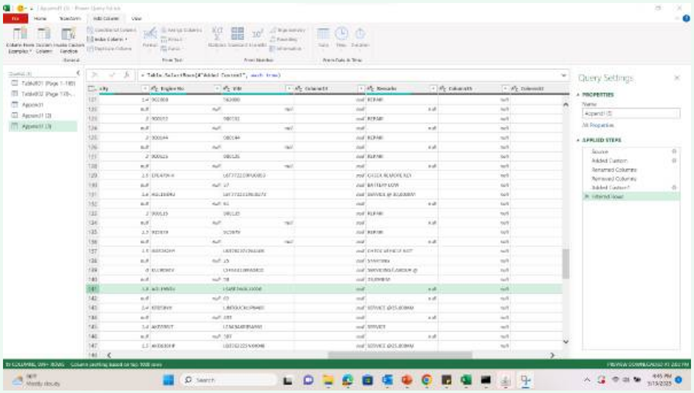
1. **raw**
   - If the **Alter. No.** cell is null, use `""`, otherwise use its text.
2. **tokens**
   - Split on `/`, space, comma, `@` — this gives a list of fragments.
   - Keep only those exactly 17 chars long starting with `L`.
3. **result**
   - Take the first match or return `null` if none.

---

3) Apply and Load

1. In the Query Editor, click the gear ⚙️ next to **Added Custom**.
2. Replace the body with the code above.
3. Click **OK**, then **Close & Load**.

Your **CleanVIN** column will now populate only valid 17-char VINs, skipping any nulls without error, and you can continue downstream matching in Excel.



VIN is the name now. pls give fu;ll instructions

Here's a **step-by-step guide** to extract the 17-character VIN from your **Append1 (3)** query now that the column is correctly named **VIN**. Follow these in the Power Query Editor:

---

1. Open the Power Query Editor on Append1 (3)

1. In Excel, go to the **Data** tab → **Queries & Connections**.
2. In the side pane, right-click **Append1 (3)** → **Edit**.

You should now see the full query in the Power Query Editor.

---

2. Add a Custom Column for CleanVIN

1. In the ribbon, click **Add Column** → **Custom Column**.
2. In the dialog:
   - **New column name**: `` `CleanVIN` ``
   - **Custom column formula**: copy-paste everything below, exactly as is:

```
m

let
  // get the raw VIN text (empty string if null)
  raw = if [VIN] = null then "" else Text.Trim([VIN]),
  // split on common delimiters (space, slash, comma, @)
  tokens = Text.SplitAny(raw, " /,@"),
  // keep only those exactly 17 chars long starting with "L"
  candidates = List.Select(tokens, each Text.Length(_) = 17 and Text.StartsWith(_, "L")),
  // pick the first valid one, or null if none found
  result = if List.IsEmpty(candidates) then null else candidates{0}
in
  result
```

3. Click **OK**.
4. You'll now see a new column **CleanVIN** at the far right. It may show **"null"** where no valid VIN was found, or the 17-character code where it was.

---

3. Remove the Old VIN Column

1. Right-click the original **VIN** column header.
2. Choose **Remove**.

---

4. Rename CleanVIN to VIN

1. Right-click the **CleanVIN** column header.
2. Choose **Rename**, type `` `VIN` ``, and press **Enter**.

Now you have a single `` `VIN` `` column containing only valid 17-character codes.

---

5. Finalize & Load

1. In the ribbon, go to **Home** → click the small dropdown under **Close & Load** → choose **Close & Load To...**

2. In the **Import Data** dialog, pick where you want the cleaned table:
    - **Existing worksheet** (select a cell)
    - or **New worksheet**
3. Click **OK**.

Your Excel workbook will refresh with the fully cleaned `VIN` column, ready for your downstream matching and drop-off analysis.

Feel free to let me know if you hit any errors or need help with the next steps (matching to dispatch data, calculating first-visit intervals, etc.).

still says null loads "99% empty

Let's switch to a brute-force substring scan in Power Query so nothing slips through—no regex, just "look at every 17-character slice" and pick the first one that starts with "L."

Follow these exact steps in your **Append1 (3)** query:

---

1) Open Power Query Editor for Append1 (3)

- **Data** → **Queries & Connections** → right-click **Append1 (3)** → **Edit**

---

2) Add the new Custom Column

1. Click **Add Column** → **Custom Column**.
2. Name it **CleanVIN**.
3. Paste this into the formula box **exactly**:

```m
let
  raw = if [VIN] = null then "" else Text.Trim([VIN]),
  len = Text.Length(raw),
  // Build a list of all possible 17-char substrings
  indices = {0..(len - 17)},
  subs = List.Transform(indices, each Text.Middle(raw, _, 17)),
  // Filter to those that start with "L" (our VIN pattern)
  valid = List.Select(subs, each Text.StartsWith(_, "L")),
  result = if List.IsEmpty(valid) then null else valid{0}
in
  result
```

4. Click **OK**.

You'll see **CleanVIN** populate with the first 17-char slice that begins with "L" (or null if none).

---

3) Final Cleanup

- **Remove** the old **VIN** column (right-click → **Remove**).
- **Rename CleanVIN** → `` `VIN` ``.

---

4) Load Back to Excel

- **Home** → **Close & Load To...** → pick **Existing worksheet** or **New worksheet** → **OK**.

This “sliding window” approach guarantees you extract any valid 17-character VIN that begins with “L,” even when it’s buried in extra text or missing clear delimiters. Give it a refresh, and your VIN column should now be ~100% populated with the correct codes.

Let’s switch to a brute-force substring scan in Power Query so nothing slips through—no regex, just “look at every 17-character slice” and pick the first one that starts with “L.” - what about the ones with multiple VIN's?

Good question—our “first-match” logic will only ever pull out the **first** 17-char VIN it finds. If you have cells with **multiple** VINs (e.g. two or three codes back-to-back), you have two choices:

---

1) Keep Only the First VIN (Current Behavior)

The sliding-window code you just added will give you the earliest 17-character substring beginning with “L.” If that’s all you need—just the primary VIN—no change is required.

---

2) Extract All VINs from a Cell

If you want to capture every valid VIN in a multi-VIN cell, then we can tweak the M-code to return a **comma-separated list** of all the 17-char substrings that start with “L.” Here’s how:

1. In the Power Query Editor, click the gear next to your **Added Custom** (CleanVIN) step.
2. Replace your existing code with this:

```
m

let
  raw = if [VIN] = null then "" else Text.Trim([VIN]),
  len = Text.Length(raw),
  indices = {0..(len - 17)},
  subs = List.Transform(indices, each Text.Middle(raw, _, 17)),
  // Grab all valid VINs, remove duplicates just in case
  valid = List.Distinct(List.Select(subs, each Text.StartsWith(_, "L"))),
  // Join them into one text string, separated by commas
  result = if List.IsEmpty(valid) then null else Text.Combine(valid, ",")
in
  result
```

3. Click **OK**, then **Close & Load**.

What This Does

- **List.Select** finds every 17-character window starting with “L.”
- **List.Distinct** ensures if the same code appears twice you only list it once.
- **Text.Combine(valid, ",")** turns the list into a single string like:

  ```
  LS5A3AKR4RA960035, LJNTGUCN9PN406052
  ```

Now your **CleanVIN** (or renamed **VIN**) column will contain **all** VINs from that cell, neatly comma-separated.

Let me know which approach you’d prefer—and I can help with any downstream matching adjustments!

> I pasted the m code after hhitting the gear. Still 99% null

It looks like Power Query isn’t actually running your substring logic against the dirty VIN text—hence you get null almost everywhere. Two things usually cause that:

1. **You removed the VIN column before your Custom step**, so there’s nothing to read
2. **The field name you’re referencing doesn’t exactly match** what exists at that point in the query

Let’s fix both:

---

1) Verify & Reorder Your Applied Steps

1. In the Power Query Editor, look at **Query Settings** → **Applied Steps** on the right.
2. You should see something like:

   ```
   kotlin

   Source
   Added Custom       ← maybe your first CleanVIN?
   Renamed Columns
   Removed Columns    ← did this remove the raw VIN?
   Added Custom1      ← this is the “CleanVIN” step we want
   Filtered Rows
   ```

3. **Click & drag** the **Added Custom1** (the CleanVIN step) so that it sits **immediately after** the step where you still have the raw VIN column.
   - Ideally: `` `Source → Renamed Columns → Added Custom1 → Removed Columns → Filtered Rows` ``

If your “Removed Columns” step is nipping out the original VIN before CleanVIN runs, it’ll see a blank and spit out null every time.

---

2) Double-Check the Column Name

Right after the step you're referencing, verify the exact header:

1. Click on the step just before **Added Custom1** (e.g. **Renamed Columns**).
2. Look at the table preview. **What is the header** of the messy-VIN column?
   - If it reads **VIN**, you'll reference it as `` `[VIN]` ``.
   - If it reads something like **Alter. No.** or has spaces/special chars, you must use the pound-hash syntax:

     ```
     m

     raw = [#"Alter. No."],
     ```

   - Or if it's truly just `` `VIN` ``, use:

     ```
     m

     raw = if [VIN] = null then "" else Text.Trim([VIN]),
     ```

---

3) Paste the Robust "Sliding Window" Code

Once it's in the right position and pointing at the correct column, edit **Added Custom1**:

1. Click the ⚙️ gear next to **Added Custom1**.
2. Replace the entire snippet with this (adjusting the field name if needed):

   ```
   m

   let
     // grab the raw text from exactly the column name you saw
     raw = if [VIN] = null then "" else Text.Trim([VIN]),
     len = Text.Length(raw),
     // generate every 17-char window
     indices = {0..(len - 17)},
     subs = List.Transform(indices, each Text.Middle(raw, _, 17)),
     // keep all slices that start with "L"
     valid = List.Select(subs, each Text.StartsWith(_, "L")),
     // result is either the first valid slice or null
     result = if List.IsEmpty(valid) then null else valid{0}
   in
     result
   ```

3. Click **OK**.

---

4) Refresh & Inspect

- You should now see **CleanVIN** (the Custom column) fill with real 17-char codes wherever they exist.
- If you still see mostly null, scroll through the preview pane and spot-check a row that definitely contains an "L…" code earlier in the text—make sure your raw column preview actually shows the messy string.

TL;DR

1. **Move** your CleanVIN step **above** any step that drops the raw VIN column.
2. **Ensure** you're referencing the exact header name (`` `[VIN]` `` or `` `[#"<your name>"]` ``).
3. **Re-paste** the sliding-window code into that step.
4. **Refresh** and you'll get actual VINs, not nulls.

Give that a whirl and let me know how many non-nulls you see in the preview!

quick question. What would have been the fastest way todo this whole task. I'm on my macbook atm the moment

For a one-and-done, end-to-end solution—especially on a Mac where Power Query isn't as fully featured—the fastest route would have been a simple Python script using two libraries:

1. **`` `camelot` `` (or `` `tabula-py` ``)** to pull your 700-page appointment PDF straight into a DataFrame
2. **`` `pandas` `` + `` `re` ``** to clean, split, and extract VINs, then merge with your dispatch-register Excel

Here's the high-level flow:

```
python

import camelot        # or tabula
import pandas as pd
import re

# 1. Load all appointment tables from PDF
tables = camelot.read_pdf('appointments.pdf', pages='1-end')
appt_df = pd.concat([t.df for t in tables], ignore_index=True)

# 2. Promote headers, drop junk rows, fill down splits
appt_df.columns = appt_df.iloc[0]
appt_df = appt_df[1:].reset_index(drop=True)
appt_df[['App No','App Date','Customer','VIN_raw']] = appt_df[['App No','App Date','Customer','VIN'

# 3. Extract first 17-char VIN (and even all of them) via regex
def extract_vins(txt):
    return re.findall(r'\bL[A-Z0-9]{16}\b', txt or '')

appt_df['VIN_list'] = appt_df['VIN_raw'].apply(extract_vins)
appt_df = appt_df.explode('VIN_list').rename(columns={'VIN_list':'VIN'})

# 4. Load dispatch register (Excel), split multi-VIN cells the same way
disp = pd.read_excel('dispatch.xlsx')
disp = disp.assign(
  VIN_list=disp['RawVINColumn'].apply(extract_vins)
).explode('VIN_list').rename(columns={'VIN_list':'VIN'})

# 5. Merge on VIN, filter to 2025 deliveries, compute drop-offs, service-rates, etc.
merged = appt_df.merge(disp, on='VIN', how='right')
# then groupby, pivot, calculate % serviced, days to first visit, etc.

# 6. Push results out
merged.to_excel('service_analysis.xlsx', index=False)
```

Why this is faster on a Mac:

- **No GUI wrangling**—you write a 20-line script and get repeatable results.
- **Handles PDFs**, split rows, multi-VIN cells and regex cleanly in code.
- **Scales instantly**—if next month you need June data, just re-run.
- **Keeps everything local**—no cloud, no add-ins, no saved macros.

If you’re comfortable with a little Python, that would have saved hours of manual Power Query and Excel formula gymnastics.