GBIT Inventory — Ninox Mac App: Minimal, Mac-Safe Build (References-Only)
==========

Goal

Avoid return-type mismatches by using **only record references (IDs)** for joins and comparisons.
`Lot Numbers` is now selected per *Intake Item* (no header text), so every line has a `Lot Numbers` reference.
All logic links by references: Product Names → Product Categories → Lot Numbers → Products.

Tables & Key Fields
----
1) Product Names
   • Product Name (text, Use as title)

2) Product Categories
   • Product Category (text, Use as title)

3) Lot Numbers
   • Lot Number (text, Use as title)

4) Intake
   • Date (optional), Notes (optional), Saved (boolean; default false)
   • Buttons: Save, Edit (toggle Saved off)

5) Intake Items (child of Intake)
   • Product Names (ref → Product Names)
   • Product Categories (ref → Product Categories)
   • Lot Numbers (ref → Lot Numbers)   ← selected per line
   • Products (ref → Products)   ← auto-linked by Save
   • Units (number)

6) Outtake
   • Date (optional), Customer (optional), Saved (boolean; default false)
   • Buttons: Save, Edit

7) Outtake Items (child of Outtake)
   • Product Names (ref → Product Names)
   • Product Categories (ref → Product Categories)
   • Lot Numbers (ref → Lot Numbers)
   • Products (ref → Products)   ← filtered picker
   • Units (number)

8) Products (the inventory index)

- Product Names (ref)
- Product Categories (ref)
- Lot Numbers (ref)
- Units On Hand (formula)
- Buttons: Delete All Products, Rebuild Products from Saved (Intakes only)

Mac-Safe Guardrails (so you don’t see “return type mismatch”)

- Resolve lot text to a **Lot Numbers record** *before* comparing/setting.
- In every `if … then … else`, both branches must return the **same type**.
- Do not call `alert()` in server blocks; we avoid `do as server` entirely.
- Link lines using references only (no name/text comparisons in final step).

MINIMAL CODE (copy into Ninox)

A) Intake → Save (button; On click)

Checks there is at least one line; validates each line has refs and Units > 0;
creates/ensures the Products row by references; links each line; sets Saved = true.

```
let me := this;
let linesA := select 'Intake Items' where Intake = me;
if count(linesA) = 0 then
  alert("Add at least one Intake Item before Save."); "Done"
else
  let created := 0; let ensured := 0; let linked := 0; let bad := 0;
  for li in linesA do
    let p := li.'Product Names';
    let c := li.'Product Categories';
    let ln := li.'Lot Numbers';
    if not p or not c or not ln or (if Units then Units else 0 end) <= 0 then
      bad := bad + 1
    else
      let pr := first(select Products where 'Product Names' = p and 'Product Categories' = c and
 'Lot Numbers' = ln);
      if not pr then
        pr := (create Products);
        pr.'Product Names' := p;
        pr.'Product Categories' := c;
        pr.'Lot Numbers' := ln;
        created := created + 1
      else
        ensured := ensured + 1
      end;
      if li.Products <> pr then
```

```
        li.Products := pr;
        linked := linked + 1
      end
    end
  end;
  if bad > 0 then
    alert("Fix " + text(bad) + " intake line(s) (missing refs or Units ≤ 0).")
  else
    me.'Saved' := true;
    alert("Intake saved. Created: " + text(created) + " | Ensured: " + text(ensured) + " | Lines
 linked: " + text(linked))
  end;
  "Done"
end


B) Outtake → Save (button; On click)
-------------------------------------
Validates lines; links each line to a matching Products row by references; does not create
Products.

let me := this;
let linesA := select 'Outtake Items' where Outtake = me;
if count(linesA) = 0 then
  alert("Add at least one Outtake Item before Save."); "Done"
else
  let linked := 0; let bad := 0;
  for lo in linesA do
    let p := lo.'Product Names';
    let c := lo.'Product Categories';
    let ln := lo.'Lot Numbers';
    if not p or not c or not ln or (if Units then Units else 0 end) <= 0 then
      bad := bad + 1
    else
      let pr := first(select Products where 'Product Names' = p and 'Product Categories' = c and
 'Lot Numbers' = ln);
      if pr then
        if lo.Products <> pr then
          lo.Products := pr;
          linked := linked + 1
        end
      else
        bad := bad + 1
      end
    end
  end;
```

```
  if bad > 0 then
    alert("Fix " + text(bad) + " outtake line(s) (missing refs, Units ≤ 0, or product not
found).")
  else
    me.'Saved' := true;
    alert("Outtake saved. Lines linked: " + text(linked))
  end;
  "Done"
end


C) Products → Units On Hand (formula field)
-----------------------------------------
let ins := sum((select 'Intake Items' where Products = this).Units);
let outs := sum((select 'Outtake Items' where Products = this and (if Outtake then
Outtake.'Saved' else false end) = true).Units);
ins - outs


D) Products → Rebuild Products from Saved (Intakes only) (button; On click)
-------------------------------------------------------------------------
let processed := 0; let created := 0; let ensured := 0; let linked := 0;
for li in (select 'Intake Items' where (if Intake then Intake.'Saved' else false end) = true) do
  processed := processed + 1;
  let p := li.'Product Names';
  let c := li.'Product Categories';
  let ln := li.'Lot Numbers';
  if p and c and ln then
    let pr := first(select Products where 'Product Names' = p and 'Product Categories' = c and
'Lot Numbers' = ln);
    if not pr then
      pr := (create Products);
      pr.'Product Names' := p;
      pr.'Product Categories' := c;
      pr.'Lot Numbers' := ln;
      created := created + 1
    else
      ensured := ensured + 1
    end;
    if li.Products <> pr then
      li.Products := pr;
      linked := linked + 1
    end
  end
end;
alert("Rebuilt from saved | processed: " + text(processed) + " | created: " + text(created) + "
```

```
| ensured: " + text(ensured) + " | lines linked: " + text(linked));
"Done"


E) Products → Delete All Products (batch; button; On click)
-------------------------------------------------------------
Requires an Admin table with boolean field “Allow Deletes”.

let batch := 100;
let admin := first(select Admin);
let prev := admin.'Allow Deletes';
admin.'Allow Deletes' := true;
let deleted := 0;
for pr in (select Products) do
  if deleted < batch then
    delete pr;
    deleted := deleted + 1
  end
end;
admin.'Allow Deletes' := prev;
alert("Deleted this run: " + text(deleted));
"Done"


F) Outtake Items → Products picker (Options filter / Constraint)
-----------------------------------------------------------------
select Products where
  'Product Names' = 'Product Names' and
  'Product Categories' = 'Product Categories' and
  'Lot Numbers' = 'Lot Numbers' and
  'Units On Hand' > 0


Quality-of-Life
---------------
• Set field-level “Writable if” on Intake Items and Outtake Items to: `not (if Intake then
Intake.'Saved' else false end)`
  and `not (if Outtake then Outtake.'Saved' else false end)` respectively.
• Product Names / Categories: lock table editing by setting “Creatable/Writable/Deletable if” to
 false; add a temporary
  “Unlock” toggle on those tables when you intentionally add new values.
• Current Inventory view: build a View element with formula
  `select Products['Units On Hand' > 0]` or filter by a selected category.
```