

ELLYPUNK

Applied

WEEKLY DIGEST / T201

31 August - 6 September 2026



This week

31 August - 6 September 2026

Available archived editions, in their original date order.

MONDAY / 31 August 2026

SQL NULLs

4

[Open original online](#)

TUESDAY / 1 September 2026

PowerShell Error Handling

9

[Open original online](#)

WEDNESDAY / 2 September 2026

Python Virtual Environments

14

[Open original online](#)

FRIDAY / 4 September 2026

Database Transactions: Commit the Whole Change

19

[Open original online](#)

SATURDAY / 5 September 2026

Windowsill Herbs: Grow Flavour, Not Sad Twigs

24

[Open original online](#)

Only retained editions are included; gaps have not been filled.

Copyright Ellypunk. Contains AI-generated artwork.

ellypunk.com

This week

31 August - 6 September 2026

Available archived editions, in their original date order.

SUNDAY / 6 September 2026

Cocktail Balance Shake Stir Taste Weekend Comic

29

[Open original online](#)

SQL NULLs

Missing Is Not Zero



A missing amount is *not* a *free* payment!

OBJECTIVE



Query *missing values* without hiding data.

THREE DIFFERENT STATES

0



Known zero

10



Known value

NULL



Value missing

WHAT NULL TELLS US



No value is recorded. The reason needs *context*.



WHY IT MATTERS



Filters, counts and averages can *surprise* you.



Keep unknowns *visible*. Check what your query *leaves out*.



NEXT: TRUE, FALSE... and UNKNOWN

MEET UNKNOWN

SQL comparisons can return a third result.

TEST: `amount = 0`



amount	result
0	TRUE
10	FALSE
NULL	UNKNOWN

WHERE KEEPS ONLY **TRUE**

FALSE and **UNKNOWN** are filtered out.

FIND MISSING VALUES

`</>`

WHERE amount **IS NULL**

FIND RECORDED VALUES

`</>`

WHERE amount **IS NOT NULL**

WATCH THE GAP

`</>`

WHERE amount `<>` 0

This also excludes **NULL**.

Use **IS NULL**, never `= NULL`, to test for missing values.



NEXT: Count rows and recorded values separately

COUNT WHAT YOU KNOW



1) COUNT ROWS

Code:

```
COUNT(*)
```

Example amounts: 0, 10, NULL

row_id	amount
1	0
2	10
3	NULL



3

rows



2) COUNT RECORDED VALUES

Code:

```
COUNT(amount)
```

Example amounts: 0, 10, NULL

row_id	amount
1	0
2	10
3	NULL



2

values



3) CHECK THE DENOMINATOR

Code:

```
AVG(amount)
```

$$(0 + 10) / 2 = 5$$

AVG ignores NULL. It does not ignore zero.



5

avg



4) DON'T INVENT A VALUE

Code:

```
COALESCE(amount,0)
```

This replaces NULL with zero.



That changes the meaning.
Use it only when the
data rule allows it.



GUARDRAIL:

Report missingness
alongside your metric



FIND THE MISSING PAYMENT



Sample data: payments	
payment_id	amount
101	0
102	10
103	NULL

1. PROBLEM



`WHERE amount = NULL`

Returns no rows.
The gap stays hidden.

2. ACTION

```
SELECT payment_id
FROM payments
WHERE amount IS NULL;
```

Result

103



3. CHECK



 3 total =
2 recorded + 1 missing

 Payment 101 still
has a real zero.

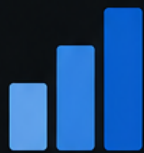
4. IMPROVEMENT

 Flag payment 103
for review.
Preserve the source.

We found
the gap
without
making up
an amount.



CHECK THE MEANING, NOT JUST THE ROW COUNT



MAKE MISSINGNESS VISIBLE



DATA CLARITY
=
BETTER DECISIONS
</>

YOUR PRACTICAL CHECKLIST

1.  Define what NULL means here.
2.  Use IS NULL or IS NOT NULL.
3.  Compare COUNT(*) with COUNT(column).
4.  Check which values AVG includes.
5.  Test zero, NULL and known values.

RECAP

-  NULL is not zero.
-  UNKNOWN is not TRUE.
-  Missing data needs an explicit rule.

TRY IT TODAY

-  Audit one nullable column. Report its missing count.



FINAL TIP FROM ELLY

Missing is a **fact** about your data, not **permission** to invent a value.



Understand the **gap**.



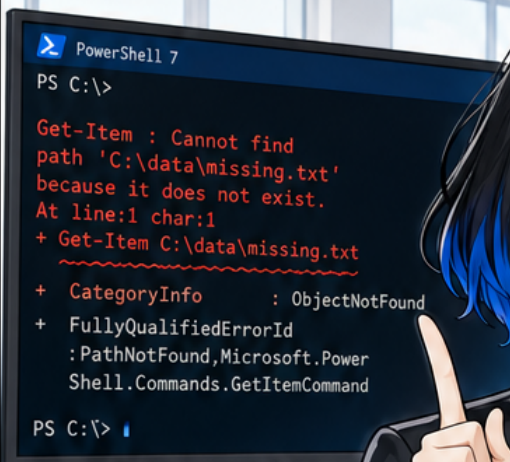
Test the **query**.



Keep the **evidence**.

POWERSHELL ERROR HANDLING

STOP • CATCH • VERIFY



FAIL FAST



HANDLE SMART



VERIFY OUTCOME

OBJECTIVE

Turn failures into clear, controlled outcomes.

A red message is not a recovery plan. Decide what must stop, what can recover, and what proves success.

STOP

MAKE FAILURE CATCHABLE

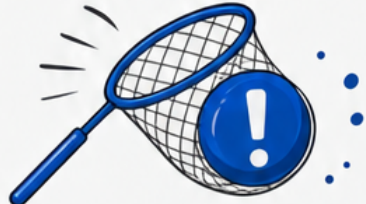


```
$ErrorActionPreference = 'Stop'
Get-Item 'C:\data\missing.txt'

# Failure becomes catchable
# Control stays with you
```

CATCH

HANDLE OR RE-THROW



```
try {
    Get-Item 'C:\data\file.txt'
}
catch {
    Write-Warning $_.Exception.Message
    # Handle it or re-throw
    throw
}
```

VERIFY

CHECK THE RESULT



```
if (Test-Path $path) {
    Write-Host "Success!" -ForegroundColor Green
}
else {
    throw "Expected file not found."
}
```



Reliable scripts **fail loudly,**
clean up safely, and **prove success.**

THE CORE LOOP

1 TRY THE RISKY WORK

```
> try { Get-Item $path -ErrorAction Stop }
```



Use `-ErrorAction Stop` when a cmdlet failure must reach catch.

2 CATCH THE ERROR

```
> catch { Write-Error $_; throw }
```



Handle an expected failure—or re-throw it.

3 FINALLY CLEAN UP

```
> finally { Remove-Item $temp -ErrorAction SilentlyContinue }
```



`finally` runs whether the try block succeeds or fails.

TRY defines the boundary.
CATCH decides the response.
FINALLY releases resources.

TRY → CATCH → FINALLY

GUARDRAILS THAT MATTER

Catch only what you
can explain or recover
from. Re-throw the rest.

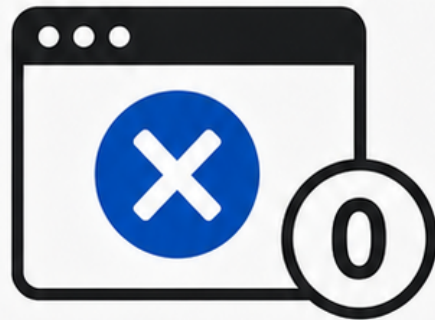
1 CMDLETS

-ErrorAction Stop makes
non-terminating cmdlet
errors catchable.



2 NATIVE TOOLS

Check \$LASTEXITCODE.
Non-zero usually means failure;
the tool defines the exact codes.



3 KEEP CONTEXT

Record the command, target,
time, and error.
Never expose secrets.



4 DO NOT HIDE SURPRISES

Use SilentlyContinue only
for an explicitly
optional cleanup.



Fail clearly. Preserve evidence. Protect data.



WORKED EXAMPLE: SAFE FILE PROMOTION



1

PROBLEM

Directly replacing a report can leave bad output.

2

ACTION

Copy to `$dest.partial` with `-ErrorAction Stop`.

3

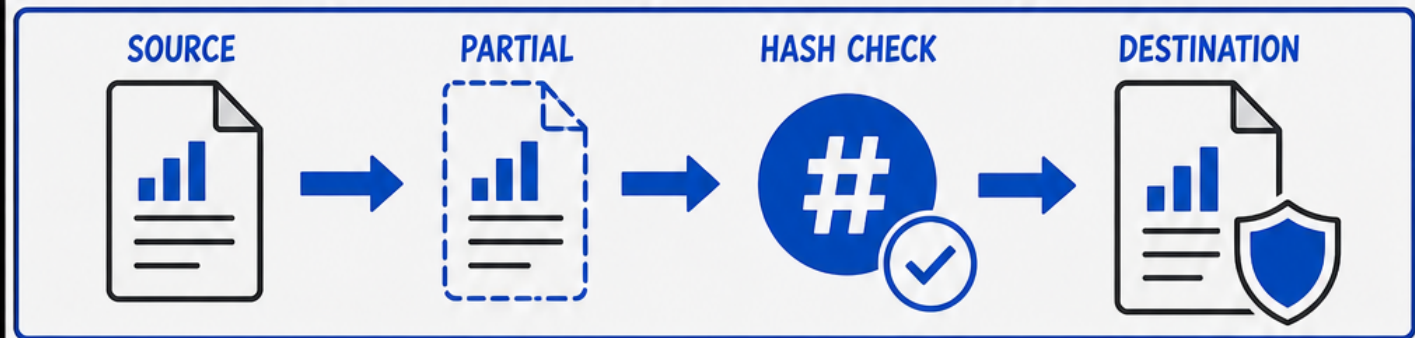
CHECK

Compare source and partial `Get-FileHash` values.

4

IMPROVEMENT

Promote only the verified file. On failure, delete the partial file and re-throw.



The old report stays *usable* until the new copy *passes* its check.



RECAP: FAIL WITH PURPOSE

ELLY'S CHECKLIST

- ☒ Set a clear failure boundary.
- ☒ Use `-ErrorAction Stop` where needed.
- ☒ Catch expected failures.
- ☒ Re-throw unexpected failures.
- ☒ Clean up temporary resources.
- ☒ Check exit codes and outputs.



REMEMBER



Implementation is not success until the result is **verified**.

FINAL TIP FROM ELLY

Success is evidence, not silence.



CATCH



VERIFY

PYTHON **VIRTUAL** ENVIRONMENTS

PAGE 1

ONE PROJECT, ONE SANDBOX



OBJECTIVE

Create, use, check, and rebuild a `.venv`.

THE PROBLEM



Project A
needs package **1.x**.



Project B
needs package **2.x**.



One global install
= **conflict**.

THE SOLUTION



Project A



`.venv`



package **1.x**



Project B



`.venv`



package **2.x**



Each project gets its
own **sandbox**.

NEXT: Build it step by step.



BUILD THE SANDBOX

PAGE 2

Five commands.
One *isolated* project.

1 CHOOSE PYTHON



```
python3 --version
```



2 CREATE



```
python3 -m venv .venv
```



3 ACTIVATE



```
macOS / Linux  
source .venv/bin/activate
```



4 INSTALL



```
python -m pip install requests
```



5 CONFIRM



```
python -c "import sys; print(sys.executable)"  
python -m pip show requests
```



Use `python -m pip` so pip follows the active interpreter.



NEXT: Guardrails that prevent drift.



FOUR RULES THAT PREVENT DRIFT

Treat the environment as **disposable**, not precious.



1 DON'T COMMIT .venv



Add `.venv/` to `.gitignore`.

2 RECORD DEPENDENCIES



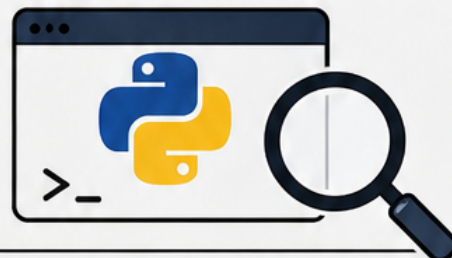
```
python -m pip freeze > requirements.txt
```

3 RECREATE, DON'T COPY



Virtual environments contain **machine-specific paths**.

4 VERIFY THE INTERPRETER



```
python -c "import sys; print(sys.executable)"
```

- ✓ Activation changes **PATH**.
- ⚠ It does **not** make the folder portable.

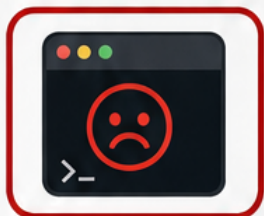


NEXT: Fix a real package failure.

WORKED EXAMPLE: FIX A MISSING PACKAGE



1 PROBLEM



```
ModuleNotFoundError:  
No module named 'rich'
```

2 ACTION



```
python3 -m venv .venv  
source .venv/bin/activate  
python -m pip install rich
```

3 CHECK



```
python -c "import rich, sys; print(sys.executable)"
```

RESULT

```
.venv/bin/python
```

4 IMPROVEMENT



```
python -m pip freeze > requirements.txt
```

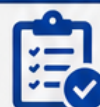
Rebuild test:

```
python -m pip install -r requirements.txt
```

Problem isolated. Result checked. Setup repeatable.



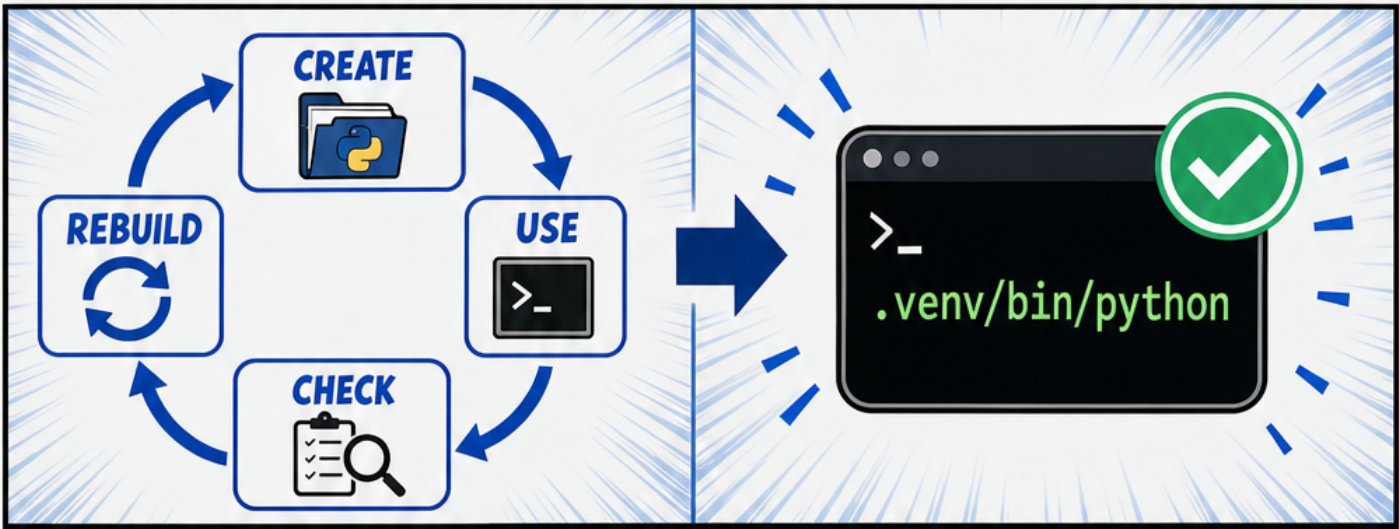
➔ **NEXT:** Your practical checklist.



YOUR .venv CHECKLIST



1	CREATE		<code>python3 -m venv .venv</code>
2	ACTIVATE		or call <code>.venv/bin/python</code>
3	INSTALL		Use <code>python -m pip</code> .
4	RECORD		Save dependencies.
5	REBUILD		Prove the setup is repeatable.



FINAL TIP FROM ELLY

-  **Trust the interpreter path, not just the shell prompt.**
-  **Rebuild from the dependency file before delivery.**
-  **One project. One sandbox. Fewer surprises.**

YOU'VE GOT THIS—BUILD REPEATABLE, RELIABLE PROJECTS! ☆

DATABASE TRANSACTIONS

PAGE 1

Commit the Whole Change

Elly: "Two updates.
One promise: both changes
succeed together."

TODAY'S OBJECTIVE



Move stock between two bins without losing units.



BEFORE

Bin A: 10

Bin B: 0

MOVE 3 UNITS

MOVE 3 UNITS

AFTER

Bin A: 7

Bin B: 3

THE RISK



If only A changes,
three units vanish
from the records.

THE FIX



Group both updates
in one transaction.



A transaction makes this database
change **all-or-nothing**.



SQLite example • Linux / macOS / Windows

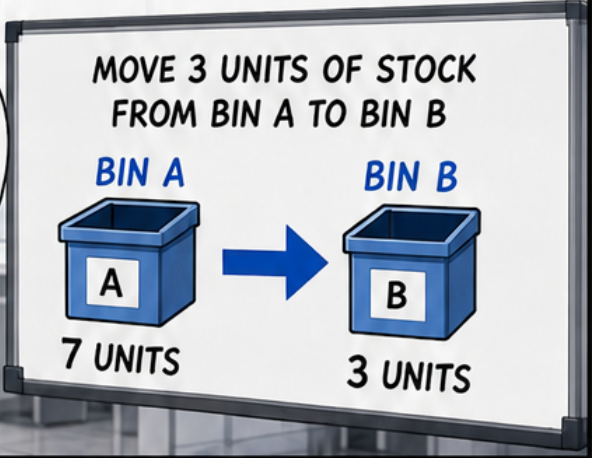
PAGE 2



DRAW THE TRANSACTION BOUNDARY

One business action can need several SQL statements.

MOVE 3 UNITS OF STOCK FROM BIN A TO BIN B



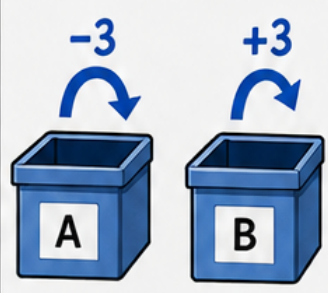
1 START

Open one transaction on one connection.



2 CHANGE

Subtract from A.
Add to B.



3 CHECK

Both rows exist.
Each update affects one row.

A = 7, B = 3,
total = 10.

Bin	Quantity
A	7
B	3
Total	10

4 FINISH

Checks pass:
commit.


COMMIT


ROLL BACK

WHAT ATOMIC MEANS

Either both updates are committed,
or neither is.



KEEP IT SHORT

Do not wait for user input or
network calls in a production
transaction.



A database rollback **cannot** unsend an email.



DATABASE TRANSACTIONS:

COMMIT THE WHOLE CHANGE



PREPARE A SAFE PRACTICE TABLE

- Use a disposable database in an existing SQL editor.
- Keep one connection. Turn off automatic transaction wrapping.
- Run this setup once, with no transaction already open.

SQLite 3 SQL • Linux / macOS / Windows

```
CREATE TABLE bins (  
  name TEXT NOT NULL PRIMARY KEY,  
  qty INTEGER NOT NULL CHECK (qty >= 0)  
);
```

```
INSERT INTO bins VALUES  
  ('A', 10), ('B', 0);
```

name	qty
A	10

name	qty
B	0

THREE GUARDRAILS



Unique names
identify the two bins.



Quantities must
exist and stay
non-negative.



The application
still checks the
transfer rules.



Constraints catch
bad values. They do not
choose your business action.

PAGE 4

MOVE THREE UNITS TOGETHER

PROBLEM

Separate commits can leave only half a transfer.

ACTION

Same connection and practice table from Page 3.
SQLite 3 SQL • Linux / macOS / Windows

```
BEGIN;
```

```
UPDATE bins SET qty = qty - 3  
WHERE name = 'A';
```

```
UPDATE bins SET qty = qty + 3  
WHERE name = 'B';
```

```
SELECT name, qty FROM bins  
ORDER BY name;
```

CHECK BEFORE FINISHING

Expected rows:

A		7
B		3

Each update must affect
exactly one row.

IMPROVEMENT

Only if every check passes, run:

```
COMMIT;
```

Otherwise stop and run:

```
ROLLBACK;
```



Run **one** finish
command, never both.

PROVE THE FAILURE PATH



PRACTICAL CHECKLIST

- ☒ One connection for the whole transaction.
- ☒ Check affected rows and business rules.
- ☒ Stop on the first error.
- ☒ Confirm commit succeeds.
- ☒ Keep external side effects outside this promise.

TRY A ROLLBACK

In a fresh practice database, repeat Page 3.

Start Page 4, but stop after its first update.

SQLite 3 SQL • Linux / macOS / Windows

```
ROLLBACK;
```



Read the bins again:

A
10

B
0

A = 10, **B** = 0.



DO NOT ASSUME

- A failed SQL statement may leave earlier changes pending.
- Roll back the active transaction before retrying.



FINAL TIP FROM ELLY

“Test what happens halfway through. That is where atomicity earns its keep.” ☆



WINDOWSILL HERBS

GROW FLAVOUR, NOT SAD TWIGS



WEEKEND WITH ELLY

Three pots.
One sunny sill.
Dinner upgraded.



OBJECTIVE:

Build a bright
little herb trio
you can actually
harvest.

BASIL



Warm, bright and
steadily moist.

CHIVES + THYME



Easy snips,
different thirst.



The trick is learning
each plant, not inventing
one watering day for
all of them.



START WITH **LIGHT**, THEN **ROOTS**

PAGE 2

1 FIND THE LIGHT



Choose your **brightest warm sill**.
Keep leaves off **cold glass**.

2 PICK THREE



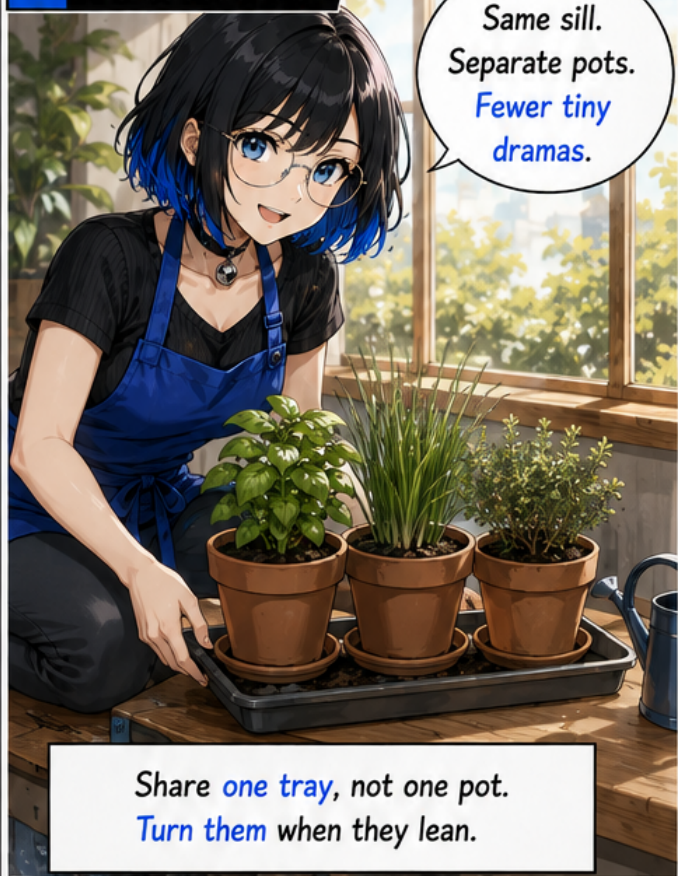
Try **basil, chives and thyme**—
useful, compact and easy to compare.

3 POT SEPARATELY



Give each herb **drainage holes**,
a **saucer** and **loose potting mix**.

4 PLACE SMART



Same sill.
Separate pots.
**Fewer tiny
dramas.**

Share **one tray**, not one pot.
Turn them when they lean.

THE KEEP-ALIVE RULES

PAGE 3

1 LIGHT



Use the **brightest warm sill**.
Turn pots for even growth.

2 WATER



Finger-test first.
Keep **basil moist**; let **thyme dry slightly**.

3 DRAIN



Water until a little **drains**.
Empty the saucer.

4 HARVEST



Snip basil just **above a leaf pair**.
Take **perennial herbs** lightly.



Feed lightly. Too much can weaken flavour.

WORKED RESCUE: FLOPPY BASIL

PAGE 4

PROBLEM

Tall, sparse,
yellowing
basil.
Wet mix.
No drainage.



ACTION

Pot with holes.
Fresh loose mix.
Pinch above
leaf pairs.
Bright warm sill.



CHECK

No standing
water.
Upright stems.
New side
shoots.



IMPROVE

Harvest little
and often,
from several
stems.



YOUR HERB-SILL CHECKLIST

PAGE 5

Tiny garden.
Proper flavour.
No allotment
required.



Bright, warm window



Separate pots with holes



Finger-test before watering



Turn pots when they lean



Snip a little, often



START WITH THREE

Learn their **tells**
before adding more.



ELLY'S FINAL TIP

The pot is tiny.
Your **pesto** ambitions
needn't be.



COCKTAIL BALANCE

SHAKE • STIR • TASTE

Great drinks
are built,
not guessed.



PAGE 1 / 5

BUILD THE BALANCE



BASE —
the backbone



SOUR —
brightness



SWEET —
rounds
the edges



DILUTION —
brings it
together



Measure.
Taste.
Change one thing.



SHAKE OR STIR?

SHAKE:

citrus, juice, cream

Chills • mixes • aerates



STIR:

clear, spirit-forward drinks

Chills • mixes • controls dilution

THE CLEAN SHAKE



TASTE, THEN TUNE



Too sharp?
Add a little sweet.



Too sweet?
Add a little sour.



Too strong?
Add dilution.



WRITE DOWN
THE WINNER



Measure your pour.
Alcohol-free works too.