// MODULE · DATA ENGINEERING

# Garbage in, lies out.

Eight vendors, the point-in-time pipeline, six corporate actions, five storage formats, and twelve checks every backtest needs.

| 08 | 05 | 06 | 12 |
|---|---|---|---|
| VENDORS | STORAGE | ACTIONS | QA CHECKS |

NEXURAL · SAGE IDEAS LLC
PUBLISHED 2026 · PERPLEXITY COMPUTER
nexural.com/learn/automation/data-engineering

// table of contents

# What you will learn.



Each section is self-contained. If you only have time for one, read section 02 — the point-in-time pipeline is the single largest source of bias in retail backtests.

// section 01

# Eight vendors. Honest notes.

The vendor you pick decides whether your backtest is research or fiction. The single most important attribute is point-in-time (PIT) accuracy — does the vendor preserve delisted symbols, store earnings by report date (not period-end), and version fundamentals so you can reconstruct 'what was known' at any timestamp?

Below: eight vendors that retail and small-fund quants actually use. Coverage, price, PIT quality, and best use case. Click any URL to visit the vendor.

| Vendor | Coverage | Price | PIT | Best for |
|---|---|---|---|---|
| Polygon.io | US equities + options tick, real-time, since 2003 | $29–$1,999/mo | Partial PIT | Retail to mid-frequency US strategies needing real-time. |
| Databento | MBO/MBP tick, all US exchanges, since 2018 | $0.10–$5/GB usage | Full PIT | Microstructure research and HFT backtesting. |
| Norgate Data | US/AU/CA equities EOD with delisting history, since 1980s | $30–$80/mo | Full PIT | Daily-bar backtests that need survivorship-free history. |
| CRSP | US equities daily + monthly, since 1925, academic gold standard | Institutional license | Full PIT | Academic research and regulatory-grade backtests. |
| IEX Cloud | US equities consolidated tape, since 2017 | $9–$499/mo | Partial PIT | Hobbyist research and low-volume production. |
| Alpaca Market Data | US equities (SIP) + crypto, real-time and historical | $0–$99/mo | No PIT | Live strategy execution where data and broker are unified. |
| Interactive Brokers | Global equities, futures, FX (TWS / API) | Subscription bundles | No PIT | Multi-asset live trading; not for serious historical backtesting. |
| Yahoo Finance | EOD bars, adjusted close, undocumented gaps | Free (unofficial) | No PIT | Tutorials and toy examples. Never production. |

// section 01 · vendor profiles 1-3

# Vendor profiles (1–3 of 8)

## Polygon.io

Coverage. US equities + options tick, real-time, since 2003

Price. $29–$1,999/mo · PIT. Partial PIT

Best for. Retail to mid-frequency US strategies needing real-time.

Visit →

## Databento

Coverage. MBO/MBP tick, all US exchanges, since 2018

Price. $0.10–$5/GB usage · PIT. Full PIT

Best for. Microstructure research and HFT backtesting.

Visit →

## Norgate Data

Coverage. US/AU/CA equities EOD with delisting history, since 1980s

Price. $30–$80/mo · PIT. Full PIT

Best for. Daily-bar backtests that need survivorship-free history.

Visit →

// section 01 · vendor profiles 4-6

# Vendor profiles (4–6 of 8)

## CRSP

Coverage. US equities daily + monthly, since 1925, academic gold standard

Price. Institutional license · PIT. Full PIT

Best for. Academic research and regulatory-grade backtests.

Visit →

## IEX Cloud

Coverage. US equities consolidated tape, since 2017

Price. $9–$499/mo · PIT. Partial PIT

Best for. Hobbyist research and low-volume production.

Visit →

## Alpaca Market Data

Coverage. US equities (SIP) + crypto, real-time and historical

Price. $0–$99/mo · PIT. No PIT

Best for. Live strategy execution where data and broker are unified.

Visit →

// section 01 · vendor profiles 7-8

# Vendor profiles (7–8 of 8)

## Interactive Brokers

Coverage. Global equities, futures, FX (TWS / API)

Price. Subscription bundles · PIT. No PIT

Best for. Multi-asset live trading; not for serious historical backtesting.

Visit →

## Yahoo Finance

Coverage. EOD bars, adjusted close, undocumented gaps

Price. Free (unofficial) · PIT. No PIT

Best for. Tutorials and toy examples. Never production.

Visit →

// section 02

# Point-in-time pipeline.

Five stages, each producing an immutable layer. Downstream stages read from the previous layer and write to a new one — never modify upstream. That immutability is what makes a backtest reproducible six months later, when you have forgotten which version of which feature you used.

The pipeline below assumes raw daily bars as input and produces a research-ready feature store. The same shape works for tick data — just substitute the parsing step.

// 01. raw ingest

# 01. Raw ingest

One-liner. Pull bytes from the vendor. Verify checksums. Never edit.

## Detail

Download raw vendor files (CSV, Parquet, FIX dumps) into an immutable bucket — S3, Backblaze, or versioned local disk. Always store the vendor's original bytes alongside the SHA256 hash and the pull timestamp. Never modify raw files; every downstream stage reads from this layer and writes to a new one.

## Output

```
raw/{vendor}/{date}.csv.gz + SHA256 manifest
```

## Common pitfall

Editing the raw file in place. If you discover a bias later, you have no audit trail. The raw layer must be append-only — this is non-negotiable in any reproducible pipeline.

// 02. normalize

# 02. Normalize

One-liner. Schema, timezones, symbols, decimals. One canonical shape.

## Detail

Parse raw into a canonical schema: same column names, same UTC timestamps, same symbol normalization across vendors. Drop or flag rows with obvious errors (negative prices, zero volume on liquid names, after-hours bars when you only want regular session). Output a versioned cleaned layer.

## Output

```
clean/{symbol}/{date}.parquet (canonical schema)
```

## Common pitfall

Silently dropping rows. Always log what you drop and why. A 0.2% drop rate during dividends usually means you removed the dividend bar — and now your strategy looks better than it should.

// 03. corporate actions

# 03. Corporate actions

One-liner. Apply splits, dividends, spinoffs — backward-adjusted.

## Detail

Build a corporate-actions table from a primary source (S&P, Norgate, CRSP). Compute a back-adjustment factor for every split and cash dividend. Apply the factor to all historical prices and volumes. Store the adjustment factor separately so you can reverse it for execution simulation.

## Output

```
adjusted/{symbol}/{date}.parquet + factors.parquet
```

## Common pitfall

Forward-adjusting (changing future prices when a corporate action happens). This contaminates today's signal with information from yesterday's open. Always back-adjust.

// 04. point-in-time align

# 04. Point-in-time align

One-liner. Reshape so any row only knows what was knowable then.

## Detail

For each row indexed at timestamp T, only include fields that were observable at T. Earnings, fundamentals, analyst estimates, index membership all become as-of timestamps. Querying 'all the data we had at 2014-06-15 09:30:00 ET' returns exactly what was true at that instant — never the latest restated value.

## Output

```
pit/{symbol}/{date}.parquet (as-of timestamps on every field)
```

## Common pitfall

Joining on a 'latest-known' fundamental table. The Compustat 2017 row was not knowable in 2014. Always join on as-of timestamp ≤ T using merge_asof or equivalent.

// 05. research-ready

# 05. Research-ready

One-liner. Features computed, cached, versioned, ready for backtest.

## Detail

Compute and cache derived features — returns, rolling statistics, factor loadings — from the PIT layer. Version the feature store (DVC, LakeFS, or a folder per feature-set hash) so a backtest run can be reproduced by referencing the feature-set version, not by re-deriving from raw every time.

## Output

```
features/{set_hash}/{symbol}.parquet
```

## Common pitfall

Not versioning. Two months later you cannot reproduce a backtest because the feature definition silently drifted. Pin the version. Hash the inputs.

// section 03

# Corporate actions.

Six corporate actions every equity backtest must handle. Each card below carries the definition, the back-adjustment formula, and a worked example with real numbers. The common thread: always back-adjust history, never forward-adjust the present.

Forward-adjustment looks innocent — you only changed prices going forward. But the day the adjustment happens, every signal your strategy computed on yesterday's data is now computed against a different yesterday. That is look-ahead bias.

// corporate action · forward split

# Forward split

SEV1 · high impact

Short form. X-for-1 (AAPL 4:1 in 2020)

## Definition

Issuer issues N new shares for every 1. Each share's price divides by N. Total market cap unchanged. Holder receives N times as many shares.

## Back-adjustment

Divide all historical prices BEFORE the ex-date by N. Multiply volumes by N. Result: a continuous, comparable price series across the split.

## Worked example

AAPL closed at $499.23 on 2020-08-28 (pre-split) and opened at $127.58 on 2020-08-31 (post-split, 4:1). After back-adjustment, pre-split prices are divided by 4: $499.23 → $124.81.

// corporate action · reverse split

# Reverse split

SEV1 · high impact

Short form. 1-for-X (small caps avoiding delisting)

## Definition

Issuer consolidates X shares into 1. Each share's price multiplies by X. Often a signal of weakness — NYSE delists sub-$1 stocks after 30 days.

## Back-adjustment

Multiply all historical prices BEFORE the ex-date by X. Divide volumes by X. Same math as forward split, inverted.

## Worked example

A 1-for-10 reverse split takes a $0.45 stock to $4.50. Historical $0.45 bars become $4.50 after adjustment. Without this fix, every 'low-price' screener silently includes a stock that was never that low post-adjustment.

// corporate action · cash dividend

# Cash dividend

SEV1 · high impact

Short form. Quarterly or special cash payout

## Definition

Issuer pays $D per share. On the ex-dividend date, the stock price drops by approximately $D at the open — the cash has left the company.

## Back-adjustment

Multiply all historical prices BEFORE the ex-date by (1 – D/close_prev). This 'total-return' adjustment ensures backtested returns include reinvested dividends. Without it, dividend-paying stocks look artificially weak.

## Worked example

MSFT pays $0.75 quarterly. On ex-date with previous close $300, the adjustment factor is (1 – 0.75/300) = 0.9975. Over 30 years, this is the difference between MSFT's 12% CAGR (price-only) and 14% CAGR (total return).

// corporate action · stock dividend

# Stock dividend

SEV2 · medium impact

Short form. Bonus shares instead of cash (e.g. 5%)

## Definition

Issuer distributes additional shares to existing holders pro-rata. Functionally similar to a small split. Price adjusts down proportionally.

## Back-adjustment

Treat as a fractional split. For a 5% stock dividend, divide pre-ex-date prices by 1.05 and multiply volumes by 1.05.

## Worked example

Reliance Industries (RIL) issued a 1:1 bonus in 2017. Effectively a 2-for-1 split. Pre-issue prices in INR were halved in the adjusted series. Failing to apply the bonus makes the 2017 price action look like a -50% crash.

// corporate action · spinoff

# Spinoff

SEV1 · high impact

Short form. Subsidiary distributed as separate ticker

## Definition

Parent distributes shares of a subsidiary to existing shareholders. Parent's share price drops by the value of the spinoff, which becomes a new tradeable ticker on day 1.

## Back-adjustment

Multiply parent's pre-ex-date prices by (parent_close_post − spinoff_value) / parent_close_post. The spinoff itself starts as a new symbol; treat it as a new instrument from day 1.

## Worked example

EBAY spun off PYPL on 2015-07-17 at a ~$36 spinoff value. EBAY closed at $61.30 the day before, opened at $25 the next day. Pre-spinoff EBAY prices need scaling by 25/61 ≈ 0.41.

// corporate action · cash merger

# Cash merger

SEV2 · medium impact

Short form. Target acquired for cash at agreed price

## Definition

Target's stock is bought out at $P/share in cash. Trading ceases at or near $P. Ticker is delisted on close date.

## Back-adjustment

No back-adjustment needed (no continuity beyond delist). But the ticker must remain in the historical universe — survivorship-free databases keep delisted tickers with their final $P payout marked.

## Worked example

Twitter (TWTR) delisted on 2022-10-27 after Musk's $54.20/share acquisition. A backtest that excludes delisted tickers misses both the merger arb opportunity and the survivorship counter-evidence.

// section 04

# Storage formats benchmarked.

Five storage formats compared on 10 years of US-equity 1-minute bars (≈60M rows × 8 columns). Benchmark machine: single M2 Mac mini reading from local NVMe. Cloud object stores add 50-200ms cold-start per read regardless of format, so columnar wins by even more on S3 / R2.

| Format | Ext | Size 1y | Size 10y | Read 1y | Read 10y |
|---|---|---|---|---|---|
| CSV (gzipped) | .csv.gz | 8.4 GB | 84 GB | 92 s | 920 s |
| Parquet (snappy) | .parquet | 1.2 GB | 12 GB | 4.1 s | 41 s |
| Arrow IPC | .arrow / .feather | 1.5 GB | 15 GB | 0.9 s | 9 s |
| HDF5 | .h5 | 1.4 GB | 14 GB | 6.5 s | 65 s |
| DuckDB | .duckdb | 1.3 GB | 13 GB | 2.4 s | 24 s |

Bottom line: Parquet is the right default for archival; Arrow IPC for hot working sets; DuckDB on top of Parquet for ad-hoc SQL exploration. CSV is fine for tiny demo files. HDF5 is mostly legacy in 2026.

// section 04 · format profiles 1-3

# Format profiles (1–3 of 5)

## CSV (gzipped) (.csv.gz)

Pros. Universal. Every tool reads it. Diffable by line.

Cons. 10× larger than columnar formats. Linear-scan reads. No types — every cell is a string until parsed.

## Parquet (snappy) (.parquet)

Pros. Columnar, typed, compressed. Predicate pushdown. Read by pandas, Spark, DuckDB, Polars.

Cons. Not human-readable. Slight write overhead vs CSV. Schema evolution requires care.

## Arrow IPC (.arrow / .feather)

Pros. Zero-copy memory layout. Fastest read in-process. Shares memory across language runtimes.

Cons. Less compressed than Parquet. Best for hot data; Parquet better for archival.

// section 04 · format profiles 4-5

# Format profiles (4–5 of 5)

## HDF5 (.h5)

Pros. Hierarchical groups. Chunked reads. Good for scientific datasets.

Cons. Single-writer locking issues. Legacy outside Python. Parquet is usually the better choice today.

## DuckDB (.duckdb)

Pros. SQL-native. Joins, aggregations, window functions in-process. Reads Parquet directly.

Cons. Single-file means single-machine. Concurrency limited. Use Parquet on disk and load into DuckDB at runtime.

// section 05

# Twelve data-quality checks.

If you cannot tick all twelve, you do not yet have evidence the strategy works — you have evidence the data passed through unchanged. Each check below has the rationale and the one-line recipe. Run them on every fresh data pull.

// category · completeness

## Completeness

### 01. Every trading day in your date range has a row

Missing days hide vendor outages and silently bias your sample.

```
pd.bdate_range(start, end).difference(df.index)
```

### 02. Every active symbol has bars for every day it was listed

Vendors sometimes drop ticks for thinly-traded names. If the symbol had volume on the tape, it should be in your data.

```
symbols_by_day = df.groupby('date').symbol.nunique()
```

// category · correctness

# Correctness

## 03. No negative prices, zero prices, or negative volume

These are data corruption or filtering artifacts. A 0 close usually means a bar with no trades was stored as 0, not NaN.

```
assert (df.close > 0).all() and (df.volume >= 0).all()
```

## 04. OHLC ordering invariants hold per row

Low ≤ Open, Close ≤ High; Low ≤ High. Violations usually mean a corporate-action adjustment was applied to some columns but not others.

```
assert (df.low <= df[['open','close']].min(axis=1)).all()
```

## 05. No prices change by more than ±50% between consecutive bars

Big jumps mean unhandled splits, stock dividends, or tick-data corruption. Real one-day crashes should be a hand-vetted small list.

```
ret = df.close.pct_change().abs(); ret[ret > 0.5]
```

## 06. Adjusted close matches dividend + split history

Compute adjusted close from raw close + corporate actions; if your vendor's adjusted close disagrees by > 1bp, one of you is wrong.

```
compute_adjusted(raw, actions) vs vendor.adj_close
```

// category · alignment

# Alignment

## 07. All timestamps are UTC with explicit timezone, not naive

Mixing naive timestamps and tz-aware ones is the #1 cause of silent off-by-N-hour bugs in research code.

```
df.index.tz is not None and str(df.index.tz) == 'UTC'
```

## 08. Trading-hour bars align to exchange calendar, not 24/7

If you include 4am or 8pm bars unintentionally, your volatility metric is computed against bars with 100× less liquidity.

```
calendar = mcal.get_calendar('XNYS')
```

## 09. Fundamentals join on as-of timestamps, never latest-known

Joining on the current Compustat row contaminates 2014 prices with 2017 restatements. As-of join is non-negotiable for PIT.

```
pd.merge_asof(prices, fundamentals, on='ts')
```

// category · survivorship

# Survivorship

## 10. Universe includes delisted symbols for the full date range

If your universe at 2015 is 'today's S&P 500', you have already biased to survivors. Use point-in-time membership.

```
universe = norgate.universe(date)  # survivors + delistees
```

## 11. Earnings are dated by REPORT date, not period-end

Q1 2024 earnings end on 2024-03-31 but are reported ~2024-04-20. Using period-end gives you 3 weeks of look-ahead.

```
earnings = df[df.filed_at <= ts]
```

## 12. Index reconstitution events tracked with as-of dates

S&P 500 adds/removes happen quarterly. Including a name from before its add-date is forward-looking; excluding it after delete-date is survivorship.

```
membership = pd.read_csv('sp500_membership.csv')
```

// section 06

# Honest pipeline checklist.

Before you ship a data pipeline into production research, tick each item below. If any one fails, the backtests downstream are not yet trustworthy.

| | |
|---|---|
| [ ] 01 | Raw vendor files are stored immutably with checksums. |
| [ ] 02 | Canonical schema applied; timezone is UTC; symbols normalized. |
| [ ] 03 | Corporate actions back-adjusted; never forward-adjusted. |
| [ ] 04 | Fundamentals joined on as-of timestamp (filed_at), not period-end. |
| [ ] 05 | Universe at every historical date includes delisted symbols. |
| [ ] 06 | S&P / index membership tracked with add and remove dates. |
| [ ] 07 | Adjusted close cross-checked against vendor; agreement within 1bp. |
| [ ] 08 | Twelve data-quality checks all pass on the latest pull. |
| [ ] 09 | Feature store is versioned; backtests reference set_hash, not raw. |
| [ ] 10 | Pipeline run is reproducible from raw to features in one command. |

Ten boxes. If you cannot tick all ten, document which you cannot tick AND why — that is the honest record.

// further reading

# Sources and further reading.

[1] Polygon.io — US market data API.

https://polygon.io

[2] Databento — institutional market data.

https://databento.com

[3] Norgate Data — survivorship-free EOD history.

https://norgatedata.com

[4] CRSP — Center for Research in Security Prices.

https://www.crsp.org/products/research-products/crsp-us-stock-databases

[5] Apache Parquet — columnar storage format.

https://parquet.apache.org

[6] Apache Arrow — in-memory columnar format.

https://arrow.apache.org

[7] DuckDB — in-process analytical database.

https://duckdb.org

[8] López de Prado — 'Advances in Financial Machine Learning' (Wiley 2018).

https://www.wiley.com/en-us/Advances+in+Financial+Machine+Learning-p-9781119482086

[9] pandas merge_asof — as-of join documentation.

https://pandas.pydata.org/docs/reference/api/pandas.merge_asof.html

[10] S&P 500 historical membership — Wikipedia compendium.

https://en.wikipedia.org/wiki/List_of_S%26P_500_companies

Continue reading → Next module: Machine Learning