ANAVSAN

PRACTITIONER'S PLAYBOOK

SNOWFLAKE COST ENGINEERING

The Complete Snowflake Cost Engineering Playbook

A practitioner's reference covering all 11 Snowflake cost domains, 93 actionable optimizations, and how to build automated governance that actually sticks.

11	93	60–70%
Cost domains covered end-to-end	Actionable optimizations with impact	of spend recoverable with right techniques

CONTENTS

11 DOMAINS

Compute · Serverless · Cloud Svc · Storage · Data Transfer · Cortex AI · SPCS · Marketplace · Governance · dbt · BI & Dashboards

WHERE YOUR MONEY GOES

All 11 Snowflake Cost Domains

Every dollar spent in Snowflake falls into one of these 11 domains. Understanding each domain's billing mechanics is the prerequisite for any cost optimization effort.

60–70%
Virtual Warehouse Compute
Per-second billing, 60s minimum. Size doubles cost each tier (XS–6XL). Burns while RUNNING, even with zero queries.

10–20%
Serverless Compute
Snowpipe, Auto-Clustering, Search Optimization, Materialized Views, Dynamic Tables, Serverless Tasks. Rates: 1.25–2 credits/compute-hr.

5–15%
Storage
$23–40/TB/month. Time Travel multiplies by up to 90×. Fail-safe adds 7 days. Clones diverge and grow.

Variable
Cortex AI
Fastest growing domain. Credits per million tokens varies by model (0.13–5.1). Warehouse credits also consumed during execution.

Free / Billed
Cloud Services
Free if <10% of daily warehouse usage. Metadata operations, query compilation, SHOW/DESCRIBE calls — invisible until expensive.

$0.02–0.04/GB
Data Transfer / Egress
Charged per GB leaving Snowflake region. Cross-cloud costs more. Ingress is FREE. Large result sets returned to client count.

Node-hr
Snowpark Container Svc
Billed while nodes ACTIVE, even idle. System CPU pool defaults to 3-day auto-suspend. GPU pools have 600s default.

Provider
Marketplace & Sharing
Auto-fulfillment compute and egress billed to provider. Consumer pays their own query compute and native app compute.

KEY INSIGHT — The Hidden Cost Multipliers

TIME TRAVEL
High-churn tables with 90-day retention can store **90×** the active data size. Set to 1 day on staging tables.

60S MINIMUM
Every warehouse resume has a 60-second minimum charge. Short-task warehouses can waste **5–10×** the actual query time.

AI TOKEN BILLING
`AI_CLASSIFY` charges **labels × rows** in tokens. Expensive models for simple tasks cost **10–50×** more.

CROSS-CUTTING — dbt & BI Workloads

dbt and BI tools span multiple domains simultaneously — they drive warehouse compute, cloud services overhead, and storage accumulation. They deserve dedicated optimization passes separate from your warehouse tuning.

See **Pages 6–7** for domain-specific recommendations for dbt models, DAG efficiency, BI refresh patterns, and metadata polling reduction.

BIGGEST COST DRIVER

Warehouse Compute — 13 Optimizations

Per-second billing with 60-second minimum per resume. Warehouse size **doubles credit consumption** at every tier. Credits burn while the warehouse is RUNNING, even with zero active queries.

```
-- Credit consumption by size (per hour)
-- XS=1 S=2 M=4 L=8 XL=16 2XL=32 3XL=64 4XL=128 5XL=256 6XL=512
SELECT warehouse_name, SUM(credits_used) total_credits,
  ROUND(AVG(avg_running), 2) avg_concurrency,
  ROUND(SUM(credits_used) * 3.0, 2) est_usd_cost
FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
WHERE start_time ⩾ DATEADD('day', -30, CURRENT_TIMESTAMP())
GROUP BY 1 ORDER BY 2 DESC;
```

PROBLEM	RECOMMENDATION	IMPACT
Idle warehouses running	Set `AUTO_SUSPEND=60s` (ETL), `300s` (interactive)	15–40% savings
Oversized warehouses	Downsize if avg execution <10s or low scan percentage	50% per tier
Underused warehouses (<5% util)	Consolidate compatible workloads onto fewer warehouses	30–60% fewer idles
Wrong scaling policy	Economy policy for batch; Standard for interactive concurrency	Avoids over-scale
Queries spilling to disk/remote	Upsize warehouse OR optimize query first	2–10× speed
Full table scans (no pruning)	Add clustering keys on filter columns	80–95% fewer partitions
No result cache utilization	Enable `USE_CACHED_RESULT`; fix non-deterministic functions	Zero credits
24/7 WH for 8-hour workloads	Schedule suspend/resume with resource monitor or task	65% savings
Runaway queries (no timeout)	Set `STATEMENT_TIMEOUT_IN_SECONDS=3600` per warehouse	Prevent runaway
Resume/suspend thrashing	Increase auto-suspend; batch tiny jobs together	No 60s penalty
No query-to-warehouse routing	Route complex queries to large WH, simple to small WH	Cost aligned
No attribution / query tagging	Implement `QUERY_TAG` standard for all pipelines	Chargeback↑
BI dashboard refresh storms	Stagger refresh schedules; dedicate a BI-specific warehouse	Peak -40–60%

D2 Serverless Compute — 10–20% of spend

PROBLEM	FIX	IMPACT
Tiny Snowpipe files (<100MB)	Batch to 100–250MB before load	30–50%↓
Over-clustering (wrong columns)	Use low-cardinality filter columns only	50–80%↓
MVs refreshing but unqueried	Drop where refresh cost > query savings	Eliminate waste
Dynamic Table TARGET_LAG too short	Set maximum acceptable staleness	10–15×↓
DT using FULL refresh mode	Set `REFRESH_MODE=INCREMENTAL`	10–100×↓
Many small serverless tasks	Consolidate with longer intervals	Less overhead
SOS on tables without lookups	Remove Search Optimization from unused tables	Eliminate waste
DTs not using DOWNSTREAM	Set intermediate DTs to DOWNSTREAM dependency	Fewer refreshes

D3 Cloud Services — Free until >10%

PROBLEM	FIX	IMPACT
CS exceeding 10% threshold daily	Reduce metadata-heavy operations, batch SHOW calls	No billed CS
BI tools polling every 30s	Client-side caching; reduce refresh intervals	Less SHOW ops
File listing overhead (large stages)	Use directory tables, partitioned external stages	Big savings
Many short-lag Dynamic Tables	Increase TARGET_LAG; use DOWNSTREAM chaining	Less scheduling

D4 Storage — 5–15% of spend

Time Travel trap: A high-churn 1TB table with 90-day retention can accumulate **90TB of Time Travel storage** — billed at full rate. Set `DATA_RETENTION_TIME_IN_DAYS=1` on staging tables immediately.

PROBLEM	FIX	IMPACT
High TT on staging tables	`DATA_RETENTION_TIME_IN_DAYS=1`	Up to 90×↓
Permanent tables for temp data	Use TRANSIENT for staging / work tables	No failsafe
Uncleared internal stages	Set `PURGE=TRUE` or run REMOVE after load	100% recovery
Abandoned tables (>90d no access)	Audit ACCESS_HISTORY and drop unused tables	Immediate
High-churn tables with failsafe	Convert to TRANSIENT + periodic backup strategy	10–16×↓
Poor clustering depth (>4)	Recluster where SYSTEM$CLUSTERING_DEPTH > 4	Less overlap
Unused MV storage (0 queries 30d)	Drop materialized views with no recent access	Reclaim storage

```
-- Find tables with large Time Travel storage
SELECT table_name, table_schema,
  ROUND(active_bytes/1e9,2) active_gb,
  ROUND(time_travel_bytes/1e9,2) tt_gb
FROM SNOWFLAKE.ACCOUNT_USAGE.TABLE_STORAGE_METRICS
WHERE time_travel_bytes > active_bytes
ORDER BY time_travel_bytes DESC LIMIT 20;
```

D5 Cortex AI Services

Fastest growing

⚠ **AI_CLASSIFY billing trap:** Charges **labels × rows** in tokens. 10 labels on 1M rows = 10M token equivalents. Minimize label count and descriptions.

PROBLEM	FIX	IMPACT
Expensive models for simple tasks	Use smaller models (mistral-7b, llama3-8b)	10-50×↓
Verbose system prompts	Minimize prompt length; efficient instructions	30-60% token cut
Large WH for AI functions	Use MEDIUM or smaller; AI is not WH-bound	4-8× WH savings
No per-user AI credit budget	Set monthly credit limits per role/user	Prevent blowouts
Row-by-row AI processing	Batch calls; process in bulk during off-peak	Throughput↑
Duplicate AI processing (same input)	Cache results for repeated identical inputs	No re-work
Over-provisioned Cortex Search	Suspend inactive search services immediately	Eliminate idle
RAG over-fetching context	Reduce retrieval K; filter relevance threshold	30-50% gen cut

D7 Data Transfer / Egress

$0.02–0.04/GB

PROBLEM	FIX	IMPACT
Cross-region COPY INTO	Use same-region staging buckets	Eliminate egress
Over-frequent cross-region replication	Reduce schedule for stable databases	Less transfer
No ECO for multi-region listings	Enable Egress Cost Optimizer — pay once	Large savings
Large result sets to client	Filter/aggregate before returning; use stages	Egress↓

D6 Container Services (SPCS)

Per node-hour

⚠ **System CPU pool default:** Auto-suspend is **3 DAYS**. An idle notebook that runs for 72 hours will be billed for all 72 hours at full node rate. Reduce to 600 seconds immediately.

PROBLEM	FIX	IMPACT
Pools running 24/7	Set `AUTO_SUSPEND_SECS`, `MIN_NODES=0`	Active billing
GPU for non-GPU workloads	Use CPU instance families instead	5-10× savings
Notebooks left running idle	Idle timeout policy + user education	Kill idle cost
System CPU pool 3-day suspend	Reduce to `AUTO_SUSPEND_SECS=600`	No 3-day idle bill
Persistent services vs. jobs	Convert long-running services to jobs	Per execution
Over-provisioned MAX_NODES	Set based on actual observed peak concurrency	Prevent blowout

```sql
-- Detect Cortex AI spend by function
SELECT query_tag, COUNT(*) calls,
  SUM(credits_attributed_compute) credits
FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
WHERE query_text ILIKE '%ai_%'
  AND start_time ⩾ DATEADD('day',-7,NOW())
GROUP BY 1 ORDER BY 3 DESC;
```

PIPELINE & ANALYTICS LAYER

dbt & BI Dashboard Cost Patterns

D9 dbt Workloads — Cross-cutting

PROBLEM PATTERN	RECOMMENDED FIX	IMPACT
Full refresh on large models every run	Convert to `config(materialized='incremental')`	80–95% less compute
DAG over-materialization	Use ephemeral models; reduce intermediate tables	Storage+compute↓
Duplicate transformations across models	Consolidate shared logic into a single base model	Less redundancy
Cascade rebuild on parent model change	Use incremental + `is_incremental()` guard	No cascade
dbt tests running on prod warehouse	Route test targets to smaller dedicated WH	50%+ WH savings
SCD2 snapshots running too frequently	Reduce frequency/scope to necessary tables only	Less compute
dbt jobs conflicting with BI refresh peak	Offset dbt schedule from BI refresh windows	Less contention

D10 BI & Dashboard Workloads — Cross-cutting

PROBLEM PATTERN	RECOMMENDED FIX	IMPACT
Dashboard refresh storms at top of hour	Stagger refresh schedules across time slots	Reduce peak 40–60%
Tableau / Power BI extract refreshes repeating	Switch to live connection where data freshness allows	Data pulled↓
BI tool metadata polling every 30s	Connection pooling; increase poll intervals	Less Cloud Svc
Low cache hit rate on dashboards	Fix dynamic `NOW()` / `CURRENT_TIMESTAMP` in filters	Cache hits↑
Oversized dedicated BI warehouse	Right-size WH; monitor utilization% per dashboard	Right-sized
No BI query tagging or attribution	Tag BI queries via `QUERY_TAG` per dashboard/team	Chargeback↑

Quick win: Most BI cost overruns trace to three causes — refresh storm scheduling, extract-mode tools pulling full tables, and dynamic timestamp filters breaking result cache. Fix these three first for immediate savings.

SENIOR PRACTITIONER LAYER

Advanced Query Anti-Patterns & Governance

D11 Advanced Query & Modeling — 10 patterns

ANTI-PATTERN	ROOT FIX	IMPACT
`SELECT *` on wide tables with 100+ columns	Select only required columns explicitly in every query	30–80% scan↓
VARIANT / OBJECT parsed at query time	Flatten JSON at ingest; persist as typed columns	2–5× faster
Window functions over very large partitions	Tighten `PARTITION BY` scope; pre-filter where possible	Proportional↓
Micro-partition skew on large fact tables	Rebuild table with even distribution key	Better pruning
Mixed ETL / BI / ad-hoc on one warehouse	Isolate by workload class into dedicated warehouses	No contention
Identical queries missing result cache	Standardize patterns; remove dynamic timestamps	Zero credits
Shadow IT / unowned pipeline resources	Audit via `QUERY_HISTORY`; assign ownership tags	Governance↑
Unintended Cartesian product / cross join	Add explicit JOIN conditions; validate with EXPLAIN	Can 10–100× compute
Correlated subquery executing per row	Rewrite as explicit JOIN or CTE with single pass	2–10× faster
Warm credits >20% of total consumption	Right-size `AUTO_SUSPEND` per arrival pattern	15–20% savings

ATTRIBUTION & CHARGEBACK FRAMEWORK

→ Tag every pipeline with `QUERY_TAG` to enable team-level chargeback

→ Map warehouses to cost centres using resource monitor labels

→ Build a workload ownership registry before any automated governance

→ Set resource monitors on *all* warehouses — including development

→ Review credit consumption weekly; act on top-5 cost drivers each sprint

Governance rule: Every warehouse should have an owner, a cost centre tag, and a resource monitor. If any of the three is missing, cost attribution breaks down immediately.

SHADOW IT & ORPHAN RESOURCE DETECTION

→ Use `ACCOUNT_USAGE.ACCESS_HISTORY` to detect orphaned resources

→ Query `WAREHOUSE_METERING_HISTORY` for warehouses with zero queries but non-zero credits

→ Identify tables with no reads in 90+ days; archive or drop after review

→ Track role-level spend: personal roles in prod are a shadow IT signal

→ Automate weekly orphan report via Snowflake Tasks to Slack/email

Common finding: In most Snowflake accounts, 15–30% of active warehouses have no identifiable owner. Governance tooling like APEX surfaces these within hours of connection.

MASTER REFERENCE

Optimization Count by Domain

13	12	5	8	12	43
Warehouse Compute	Serverless Compute	Cloud Services	Storage	Cortex AI +Advanced	SPCS · dbt · BI · Transfer · MV

HIGHEST-IMPACT COMPUTE WINS — 5 PICKS

Idle WH running between jobs	`AUTO_SUSPEND=60s` for batch WH	15-40%↓
WH oversized for workload	Drop 1 size tier if avg exec <10s	50% per tier
Full table scan (no clustering)	Add cluster key on filter columns	80-95%↓
Same query not cache-hitting	Fix dynamic timestamps/functions	Zero credits
24/7 WH for scheduled jobs	Schedule suspend/resume via task	65% savings

STORAGE QUICK WINS — 4 PICKS

Staging tables with 90-day TT	`DATA_RETENTION_TIME_IN_DAYS=1`	Up to 90×↓
PERMANENT for temp/stage data	Convert to TRANSIENT tables	No failsafe
Internal stages not purged	`PURGE=TRUE` on COPY INTO	100% recovery
Abandoned tables (>90d unused)	Query ACCESS_HISTORY; drop or archive	Immediate

CORTEX AI COST CONTROLS — 4 PICKS

claude/mistral-large for simple tasks	Switch to mistral-7b or llama3-8b	10-50×↓
No per-user AI budget limits	Resource monitors by role/user	Prevent blowouts
Row-by-row AI calls in pipelines	Batch + off-peak processing	Throughput↑
Idle Cortex Search service	Suspend when not serving queries	Eliminate idle

DBT TOP OPTIMIZATIONS — 4 PICKS

Non-incremental large models	`materialized='incremental'`	80-95%↓
Over-materialized intermediates	Use ephemeral models	Less storage
dbt tests on prod warehouse	Dedicated small test warehouse	50%+ WH savings
dbt + BI running simultaneously	Offset dbt schedule by 30–60min	Reduce concurrency

SERVERLESS QUICK WINS — 3 PICKS

Tiny Snowpipe files (<100MB)	Batch to 100–250MB before load	30-50%↓
DT TARGET_LAG too short	Set max acceptable staleness	10-15×↓
DT with FULL refresh mode	`REFRESH_MODE=INCREMENTAL`	10-100×↓

SPCS EMERGENCY FIXES — 2 PICKS

System CPU pool 3-day suspend	`AUTO_SUSPEND_SECS=600`	Avoid 72hr idle bill
Notebooks left open	Idle timeout policy + user training	Kill idle cost

SENIOR PRACTITIONER TECHNIQUES

Advanced Cost Engineering Patterns

ADVANCED QUERY PATTERNS

ANTI-PATTERN	ROOT FIX	IMPACT
Redundant computation across 3+ models	Single intermediate materialization	No dup credits
Query perf regression over time	Decompose: growth vs. decay vs. contention	Stops drift
Functionally identical queries (cache miss)	Standardize patterns for cache hits	Zero credits
VARIANT/OBJECT parsed at runtime	Flatten at ingest time into typed columns	2-5× faster execution
Window functions on huge partitions	Tighten `PARTITION BY` clause	Proportional↓
Micro-partition skew on large tables	Rebuild table with even distribution	Better pruning
Warm credits >20% of total	Right-size auto-suspend per arrival pattern	15-20% savings

```
-- Find queries with partition over-scan
SELECT query_id, query_text,
  partitions_scanned, partitions_total,
  ROUND(partitions_scanned/partitions_total*100,1)
pct_scanned,
  ROUND(credits_used_cloud_services,4) credits
FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
WHERE partitions_scanned > partitions_total * 0.5
  AND start_time ⩾ DATEADD('day',-7,NOW())
ORDER BY credits DESC LIMIT 25;
```

CORTEX AI ADVANCED (8 PATTERNS)

PROBLEM	FIX	IMPACT
RAG over-fetching context	Reduce retrieval K; filter relevance threshold	30-50% tokens↓
Embeddings regenerated for unchanged data	Cache embeddings; regenerate only on change	No reprocessing
Cortex Analyst model too broad	Scope to relevant tables/columns per question type	40-60% tokens↓
AI agents firing on noise	Add outcome validation; reduce trigger frequency	No wasted runs
Fine-tuning when few-shot works	Try prompting/few-shot first before fine-tuning	No training cost
Cortex Search over-rebuilding	Match refresh cadence to actual data churn rate	50-80% rebuild↓
Document AI reprocessing unchanged docs	Track doc hashes; skip on no change	Eliminate waste
No per-tenant AI attribution (ISV)	Implement tenant-level token tracking	Margin analysis

GOVERNANCE & ATTRIBUTION

→ Tag every pipeline with `QUERY_TAG` to enable team-level chargeback

→ Use `ACCOUNT_USAGE.ACCESS_HISTORY` to detect shadow IT and orphaned resources

→ Build a workload ownership registry before deploying any automated governance tooling

→ Set resource monitors on all warehouses — even development ones — with a weekly reset

OFTEN OVERLOOKED COST DRIVERS

Marketplace, Sharing & Resource Controls

D8 Marketplace & Data Sharing — Variable billing

PROBLEM PATTERN	RECOMMENDED FIX	IMPACT
Marketplace app running 24/7 warehouse	Check app WH config; set auto-suspend to 60s	Large savings
Cross-region marketplace delivery	Use same-region provider account; enable ECO	Eliminates egress
Listing with AUTO_FULFILLMENT enabled	Disable auto-fulfillment on low-demand listings	No idle compute
Provider replicating to all regions	Limit replication to regions with active consumers	Egress cost down
Shared DB queried without local cache	Materialize frequently-joined shared data locally	No remote scan
Unused marketplace trials still active	Audit and terminate inactive app trials monthly	Remove idle cost

Provider billing: As a data provider you pay for compute that powers queries on your listing. Monitor `DATA_SHARING_USAGE.LISTING_CONSUMPTION_DAILY` to see which consumers drive your provider costs.

RESOURCE MONITOR SETUP

MONITOR TYPE	CONFIGURATION	BENEFIT
Account-level monitor	Monthly credit quota; alerts at 75/90/100%	Global safety net
Per-warehouse monitor	Individual quota per WH aligned to workload	Team-level control
Dev / sandbox warehouses	Weekly reset + suspend-at-100% to hard-cap	No runaway dev
Cortex AI heavy roles	Role-level monitor on AI-heavy roles monthly	AI budget control
Alert routing	Route alerts to Slack or email via Tasks	Fast response

```
-- Account-level resource monitor
CREATE OR REPLACE RESOURCE MONITOR account_guard
  WITH CREDIT_QUOTA = 5000
  FREQUENCY = MONTHLY START_TIMESTAMP = IMMEDIATELY
  TRIGGERS
    ON 75 PERCENT DO NOTIFY
    ON 90 PERCENT DO NOTIFY
    ON 100 PERCENT DO SUSPEND;
```

Critical gap: Most accounts have no account-level monitor. A single runaway loop or recursive CTE with no STATEMENT_TIMEOUT can exhaust months of credits in hours.

WHEN THINGS GO WRONG

Cost Spike Investigation Playbook

STEP 1 — IDENTIFY THE SPIKE WINDOW

```
-- Daily credit consumption -- spot the anomaly
SELECT DATE_TRUNC('day',start_time) dt,
  ROUND(SUM(credits_used),2) daily_credits
FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
WHERE start_time ⩾ DATEADD('day',-30,NOW())
GROUP BY 1 ORDER BY 1;
```

STEP 2 — IDENTIFY THE WAREHOUSE

```
-- Which warehouse drove the spike?
SELECT warehouse_name,
  ROUND(SUM(credits_used),2) credits,
  COUNT(*) intervals
FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
WHERE start_time BETWEEN 'SPIKE_START' AND 'SPIKE_END'
GROUP BY 1 ORDER BY 2 DESC;
```

STEP 3 — FIND THE OFFENDING QUERIES

```
-- Top credit queries in the spike window
SELECT query_id, user_name, warehouse_name,
  ROUND(credits_used_cloud_services,4) credits,
  ROUND(execution_time/1000,1) exec_secs,
  LEFT(query_text,80) preview
FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
WHERE start_time BETWEEN 'SPIKE_START' AND 'SPIKE_END'
  AND warehouse_name = 'TARGET_WH'
ORDER BY credits DESC LIMIT 20;
```

STEP 4 — IDENTIFY THE USER / ROLE

```
-- Who drove spend in the anomaly period?
SELECT user_name, role_name,
  COUNT(*) queries,
  ROUND(SUM(credits_used_cloud_services),2) total
FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
WHERE start_time BETWEEN 'SPIKE_START' AND 'SPIKE_END'
GROUP BY 1,2 ORDER BY 4 DESC LIMIT 15;
```

STEP 5 — CHECK FOR RUNAWAY LOOPS

```
-- High-frequency repeated queries
SELECT LEFT(query_text,60) pattern, user_name,
  COUNT(*) executions,
  ROUND(AVG(execution_time)/1000,1) avg_secs,
  ROUND(SUM(credits_used_cloud_services),3) total
FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
WHERE start_time BETWEEN 'SPIKE_START' AND 'SPIKE_END'
GROUP BY 1,2 HAVING executions > 30
ORDER BY total DESC LIMIT 10;
```

SPIKE TYPE DECISION TREE

WHICH ACCOUNT_USAGE VIEW TO INVESTIGATE?

→ **Compute spike** — WAREHOUSE_METERING_HISTORY + QUERY_HISTORY

→ **Storage spike** — TABLE_STORAGE_METRICS + STORAGE_USAGE

→ **Serverless spike** — SERVERLESS_TASK_HISTORY + DYNAMIC_TABLE_REFRESH_HISTORY

→ **Cortex AI spike** — CORTEX_USAGE_HISTORY (token usage by function)

→ **SPCS spike** — COMPUTE_POOL_EVENTS + container node-hour billing

ANAVSAN

RUNNING THESE 93 OPTIMIZATIONS MANUALLY?

Let *APEX* detect, route, and track every one of them.

APEX monitors all 11 Snowflake cost domains in real time, automatically routes anomalies to the right owner, and generates verified savings reports — without adding headcount or engineering toil.

AI Anomaly Detection
ML-based detection across all 11 domains. Not just warehouse thresholds — storage, Cortex AI, SPCS, and serverless too.

Automated Routing
Every detected issue is mapped to a workload owner and routed with context — no manual triage required.

Verified Savings Proof
Before/after credit delta reports that hold up to CFO scrutiny. ROI dashboard built in from day one.

93 Recommendations Built In
Every optimization in this playbook is encoded as an APEX detection rule — with confidence scores and risk assessment.

dbt + BI Coverage
Model-level and DAG-level cost attribution for dbt. BI refresh storm detection and metadata polling alerts.

Live in 24 Hours
Read-only Snowflake data share. No agents. No code changes. First anomalies surface within hours of connection.

Book APEX Demo →

See APEX platform details ?