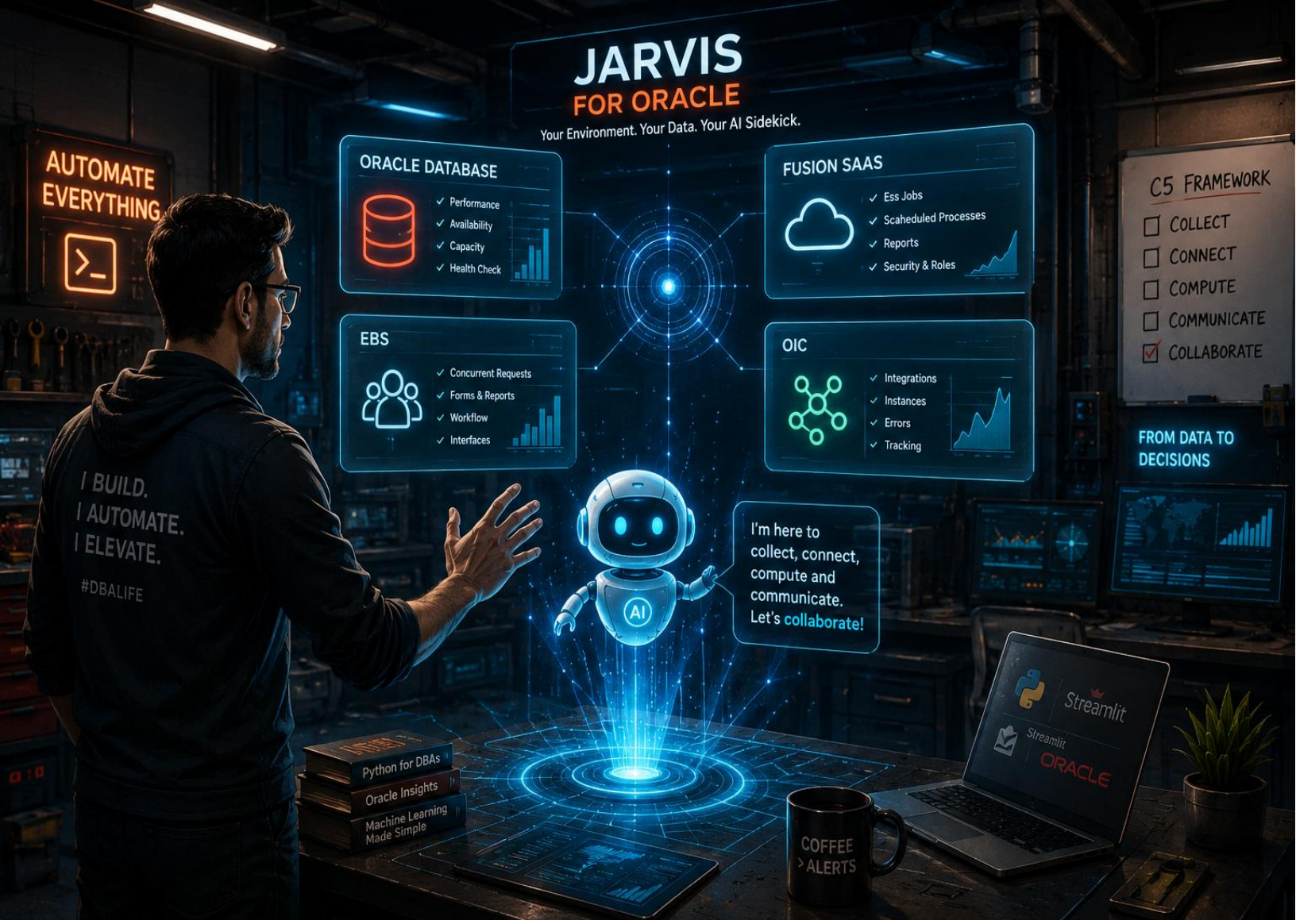


DIY- Building Your AI Sidekick Across the Oracle Stack

Akhash Ramnath
Oracle ACE Professional

DOYENSYS
VALUE INNOVATION

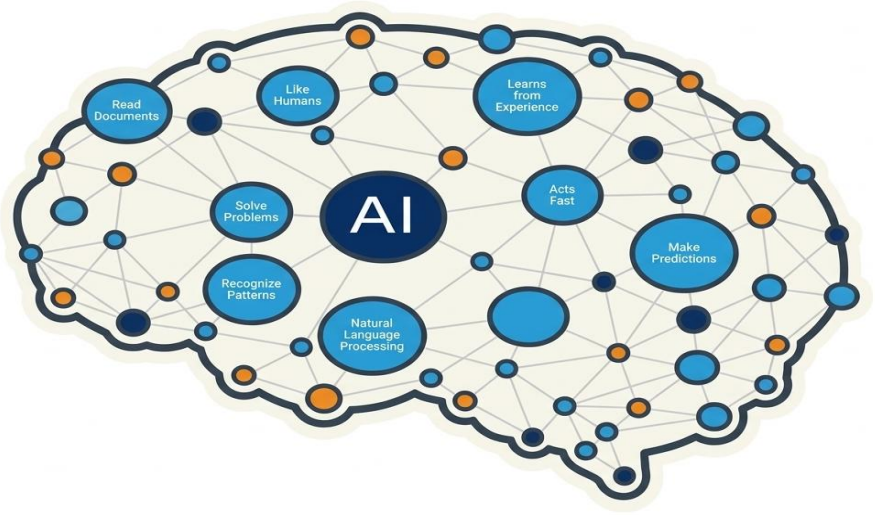
About this Write Up

Managing the full Oracle Enterprise Stack - Database, EBS, and Fusion SaaS - often means juggling scripts, dashboards, alerts, and repetitive checks. Traditional monitoring tools and native support systems help, but they rarely deliver proactive or intelligent insights. This write up shows how to build something better. We begin by examining the limitations of standard monitoring and support models, then move into a practical, Do-It-Yourself approach using Python and Machine Learning - building a custom AI sidekick tailored to your Oracle ecosystem.

> ***"Every expert was once a beginner - AI is no different. Start small, grow fast."***

1. What is AI?

Artificial Intelligence is software that analyses data to automate complex tasks, optimise performance, and predict outcomes, a programmable "digital brain" for your infrastructure and applications. In simple terms, think of it as a very fast, very tireless intern that never asks "is this in scope?"



I know, I know you've heard this a thousand times. But bear with me for 60 seconds.

AI is basically software that watches data, finds patterns, and acts on them. Like a really smart intern except it doesn't take coffee breaks, doesn't need onboarding, and never asks 'is this in scope?'

Think of it as a digital brain for your systems. It reads documents, recognises patterns, solves problems, makes predictions all without a 2 AM phone call to you.

Simple enough? Good. Let's move on before it gets philosophical.

2. It's Everywhere!

AI is not a distant concept anymore. It is already embedded in the apps and devices you use every single day, from the content your streaming platform recommends, to the cab that arrives without a phone call, to the search engine that knows what you need before you finish typing.

So how everywhere is AI, really? Let me give you some real examples.

I Googled some inspirational quotes one afternoon you know, one of those days. Next thing I know, My Instagram is flooding me with motivational reels. Fine. But then it started suggesting 'How to save your marriage' quotes. I'm not married! AI just assumed things. Bold of it.

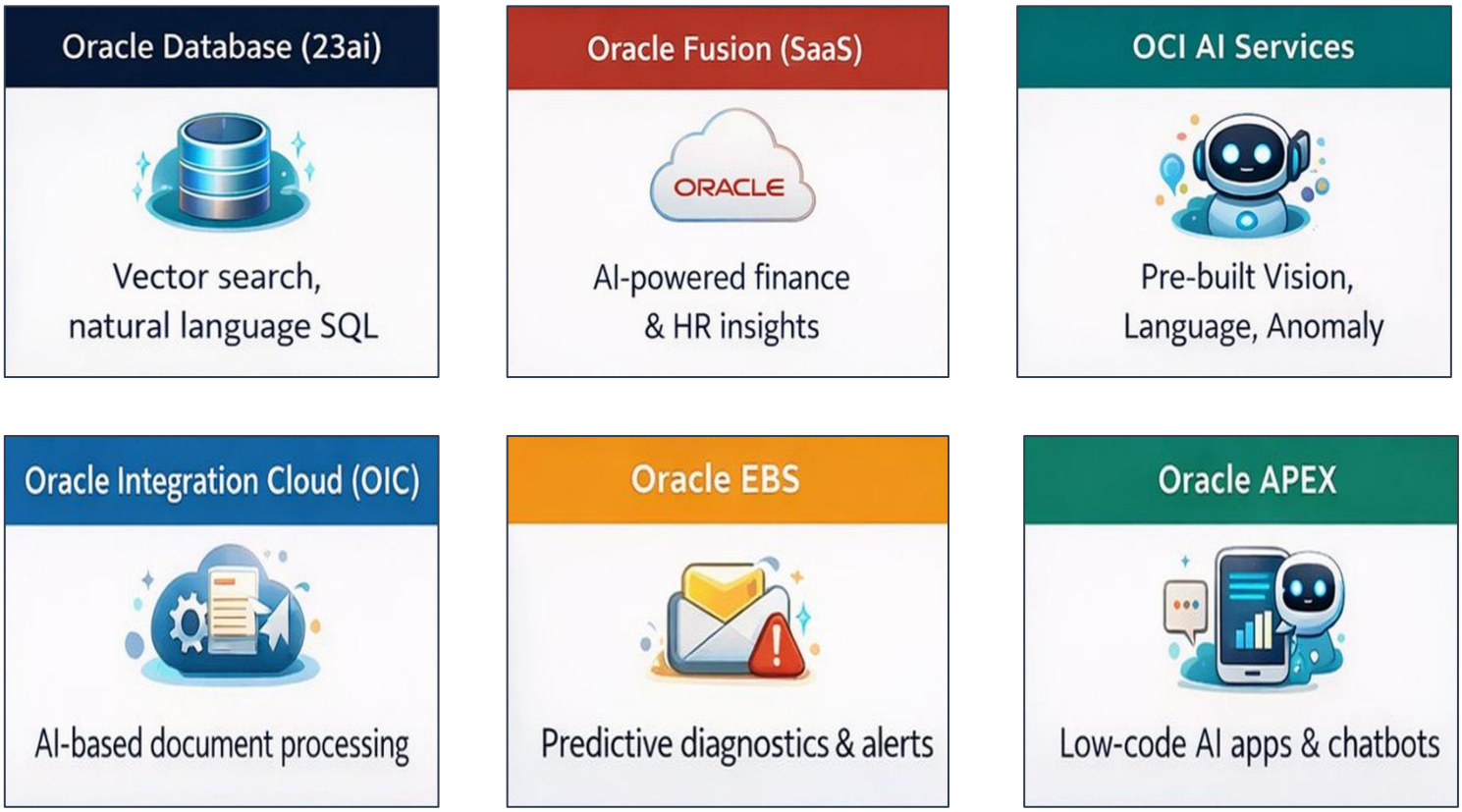
And my mom she's back in India. She used to call me every weekend: 'What movie should I watch? How do I book a cab? How do I order food?' Now? Silence. Complete silence. Netflix already suggests all her Tamil movies. Alexa books her cab. Swiggy knows her dinner order before she does.

On one hand AI is incredible. On the other hand my mom doesn't need me anymore.

That's AI. It's in your phone, your TV, your cab app, your grocery app. And yes it's also in your Oracle stack. Which is exactly what we're here to talk about.

3. AI Is Already in Your Oracle World

AI is not coming to the Oracle ecosystem it is already here. Across six key areas of the Oracle stack, intelligent capabilities are available today, actively used in production environments worldwide.



Oracle Database 23ai: Vector Search & Natural Language SQL

Instead of writing complex SQL queries, you can now ask your database questions in plain English. "Show me all employees who haven't taken leave in 6 months" the database figures out the SQL for you. It's like having Google inside your database.

Oracle Fusion SaaS: AI-powered Finance & HR Insights

With over 50 embedded AI agents across ERP, HCM, and SCM, Fusion now flags anomalies and surfaces insights automatically including a Payables Agent, Ledger Agent, and Planning Agent all available at no extra cost.

OCI AI Services: Vision, Language, Anomaly Detection

Pre-built, ready-to-use AI models you can plug into your applications. OCI Vision reads scanned invoices and extracts data directly into your system no manual keying, no data-entry marathons.

Oracle Integration Cloud: AI-based Document Processing

A vendor emails an invoice PDF. OIC picks it up, reads it, understands it is an invoice, extracts the fields, and pushes the data into EBS or Fusion automatically. Your team just watches it happen.

Oracle EBS: Predictive Diagnostics & Alerts

Natural Language Query of EBS 12.2 is now available via OCI Generative AI. For predictive diagnostics, the DIY approach this session teaches is the most practical and flexible path.

Oracle APEX: Low-code AI Apps & Chatbots

Status: LIVE & VERY ACTIVE

Build smart applications and chatbots on top of your Oracle data with minimal coding. Real customers like Munich Re HealthTech have already launched AI chatbots on APEX handling 90% of employee questions.

> ***"The ecosystem is AI-ready. Are YOU ready to use it?"***

4. Why Should I Care?

This is the question in every room when AI comes up. Will it take my job? Should I be worried? The short answer no. The longer answer you should be more worried about NOT learning it.

> *You're thinking 'Great, another AI talk. Last month my manager forwarded me an article saying AI is coming for my job.'*
>
> *I get it. The fear is real. But let me ask you something when cloud came along, remember what they said? 'We won't need DBAs anymore. Everything will be managed.' And yet here you are. Still employed. Still getting called at 2 AM. Clearly cloud didn't solve everything.*
>
> *AI is the same story but faster, and honestly, more interesting. The people who learned cloud early became the cloud architects. The people who ignored it spent three years catching up.*

> ***"AI won't replace humans. But humans who use AI will replace those who don't. - Sam Altman, CEO, OpenAI"***

The goal of this article is to make sure you are on the right side of that sentence.

---

5. Back When Consultants Were the AI

Before AI, before dashboards, before automation the intelligent system in the room was us. Experienced consultants were the monitoring tools, the alert systems, the pattern recognisers, and the root cause analysers all rolled into one very tired human being.

Let's rewind a little. Before AI, before fancy dashboards, who was the intelligent system in the room? We were.

You got an alert at 2 AM? You woke up, SSH'd into the server, scrolled through alert logs line by line manually like a detective at a crime scene with no coffee.

Something broke in production? You opened 15 browser tabs. Oracle docs, MOS, Stack Overflow, that one blog post from 2014 that somehow always had the answer.

We correlated issues across EBS, middleware, and database in our heads manually while someone from the business was sending escalation emails every 10 minutes.

We WERE the AI. Human intelligence, running on caffeine, tribal knowledge, and the quiet prayer that nothing else would break before morning. And honestly? We were good at it. But it was exhausting. And it wasn't scalable.

---

6. Imagine With an Intelligent Assistant Beside You

Now picture the same consultant with the same expertise and the same Oracle environment but with an AI sidekick beside them. Everything changes.

- AI highlights only critical alerts the noise is gone.
- Automation runs in the background you are called only when it actually matters.
- Issues are detected at 2 PM, not discovered at 2 AM.

- Ask in plain English "Why is PROD slow?" and receive the query, the data, and a recommendation.
- One unified dashboard instead of 20 tabs.
- Smart log summaries instead of scrolling through thousands of lines.
- Documentation that writes itself.

> *The expertise is still yours. The domain knowledge is still yours. AI just removes the grunt work so you can focus on what actually requires a human judgement, context, and experience.*

> ***"You're not being replaced. You're being upgraded."***

---

7. Why Build It Yourself?

With commercial monitoring tools and vendor AI products available, why take the DIY route? The answer is the same reason people love IKEA.

> *Have you ever bought furniture from IKEA? You go in. You pick a beautiful bookshelf. You bring home four flat boxes, one instruction booklet with no words just stick figures and a tiny Allen key that will test your patience and your marriage.*
>
> *Three hours later it's done. And you feel AMAZING. You built that. You understand every part of it. If a shelf wobbles, you know exactly which screw to tighten.*
>
> *That's exactly the feeling we're going for here. When you build your own AI sidekick you know what it does, why it does it, and how to fix it when it doesn't. You've tuned it for YOUR Oracle environment. Not a generic one. Yours.*

And unlike the IKEA bookshelf Python is free. The libraries are free. And there's no Allen key involved.

The core reasons to DIY:

- Full ownership you know every part of what you built.
- Tailored to YOUR environment not a generic product.
- Zero vendor lock-in.
- Pride of ownership “Oh yeah, I built that."
- Python is free. Libraries are free. Your database is already there.

---

8. How Good Do You Need to Be at Python?

This is the question that stops most Oracle professionals before they even start. The honest answer? You need to know very little to begin and the rest you look up as you go.

> *The basics take about an afternoon. Print something. Loop through a list. Connect to a database. That's genuinely all you need to get started.*
>
> *"But what about CI/CD pipelines? TensorFlow? Machine learning frameworks?" You Google that part. Seriously. That's not a joke that's the actual answer. Nobody memorises that stuff. Even senior developers Google it every single day.*
>
> *The difference between a beginner and an expert in Python is not what they know it's how fast they Google.*
>
> *And in 2025 you have something even better than Google. You have AI to help you write the code. The irony is beautiful you're using AI to build your AI sidekick.*

"Don't wait to become an expert. Build, Learn, and Improve along the way."

Start simple. Start today. The best time to write your first Python script was last year. The second best time is after this session.

---

9. Start Small. Start Today.

Four steps. That's all it takes to go from zero to a working Oracle AI dashboard in under 30 minutes, tonight, in your hotel room if you want.

Step 1: Install Python & Streamlit

pip install streamlit oracledb pandas scikit-learn oci

One command. Two minutes. You now have everything you need.

Step 2: Connect to Your Oracle System

Copy the 10-line connection script from today's GitHub repo. Paste it. You are now connected to your Oracle database from Python.

Step 3: Add Your First SQL Query

st.dataframe(select * from gv$database)

Your database identity live, on a webpage. That's your first feature.

Step 4: Run It

streamlit run app.py

Open your browser. Share the link with your team. You just built a working dashboard.

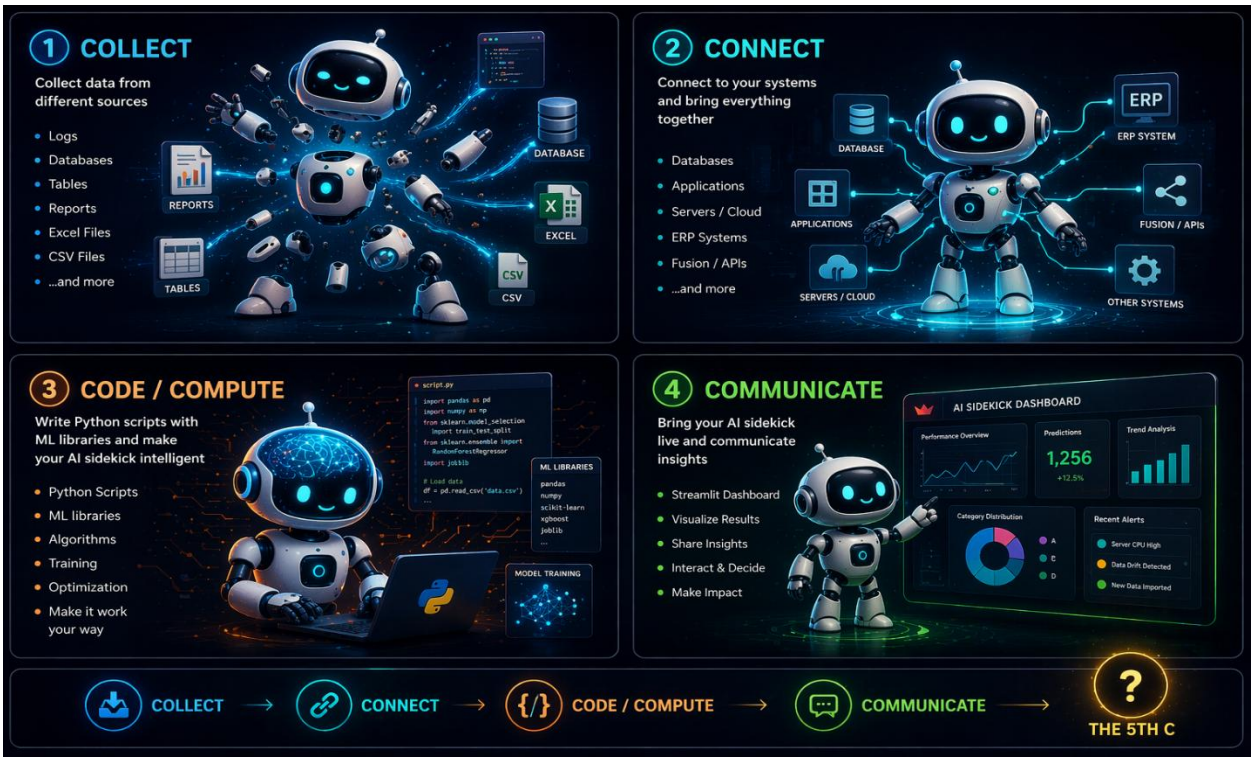
> *Next week you add a chart. Week after that you add an alert. Month later you add a prediction. That's how every great tool starts embarrassingly simple.*

> ***"You don't need permission. You don't need a budget. You just need to start."***

---

10. The C5 Framework Your Blueprint

Every good DIY project needs a plan. The C5 Framework is a five-step blueprint for building your AI sidekick structured, repeatable, and scalable across any Oracle environment.



C1: Collect

Gather your data from every source your Oracle environment produces logs, database tables, reports, Excel files, CSV dumps, AWR reports. Think of this as stocking your fridge before cooking. Your Oracle environment generates mountains of data every second. You're just not cooking with it yet.

C2: Connect

Connect to all your systems in one place database, EBS, Fusion APIs, OIC, servers. Pull everything into a single Python script that talks to your entire stack at once. Your systems have been waiting for someone to introduce them to each other.

C3: Compute / Code

Write Python scripts with ML libraries pandas, scikit-learn, numpy and make your sidekick intelligent. Detect anomalies. Predict failures. Score risk. You don't need to understand how the engine works to drive the car. You just need to know which library to call.

C4: Communicate

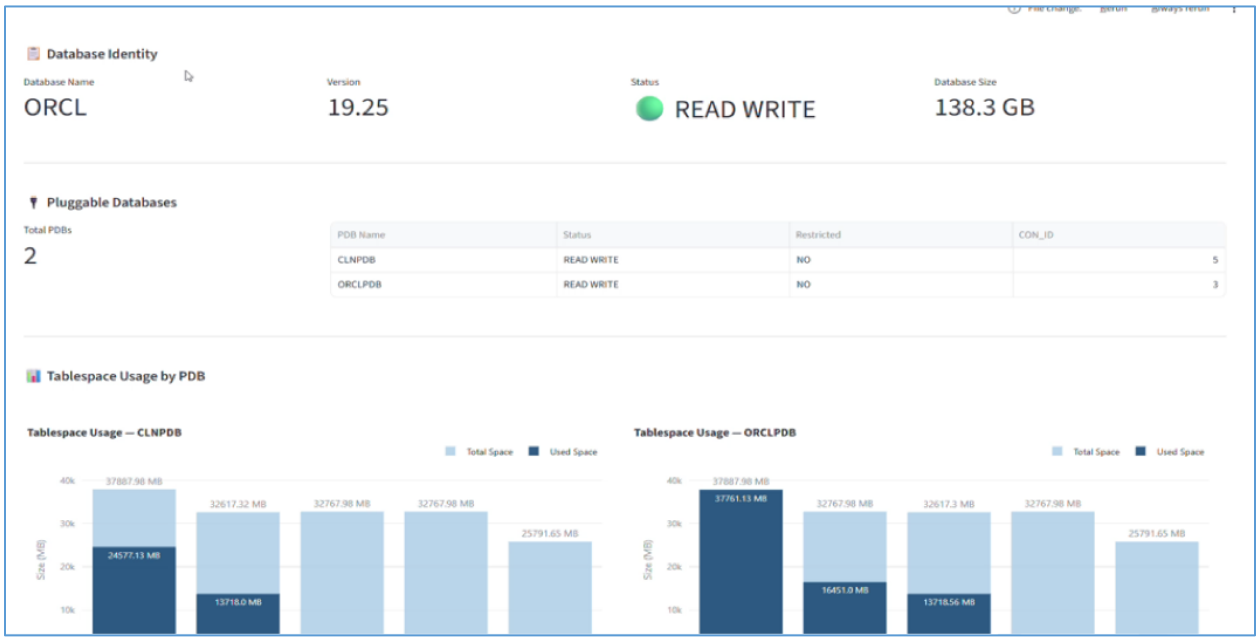
Build a Streamlit dashboard. Visualise results. Share with your team. Set up alerts. Transform your terminal script into a tool that impresses everyone not just you.

C5 ?

The fifth C is the most important one. It ties everything together. And it's being saved for the conclusion. Stay tuned.

10a. Demo 1 – DB Health Graph

The DB Health Graph is the simplest script in the sidekick toolbox. No machine learning, no AI — just Python talking directly to Oracle and turning the answers into a live, visual dashboard. Despite its simplicity, it replaces something DBAs used to do manually every single morning.



Collect

The script reads five live data sources from Oracle: database identity and version (V$DATABASE, V$INSTANCE), total database size across all datafiles (CDB_DATA_FILES), a list of all Pluggable Databases (V$PDBS), tablespace usage per PDB (DBA_TABLESPACE_USAGE_METRICS), and the largest schemas and segments (DBA_SEGMENTS). Think of these Oracle views like a hospital's vital signs monitor — the readings are always there, always live. We're just finally building a screen to display them.

Connect

One connection to the Oracle CDB as SYS/SYSDBA using the oracledb Python library — Oracle's official driver. The script uses ALTER SESSION SET CONTAINER to step into each PDB individually, like opening different rooms in the same building without needing a separate key for each one. One line of code. One handshake. Python now sees everything the database knows.

Compute

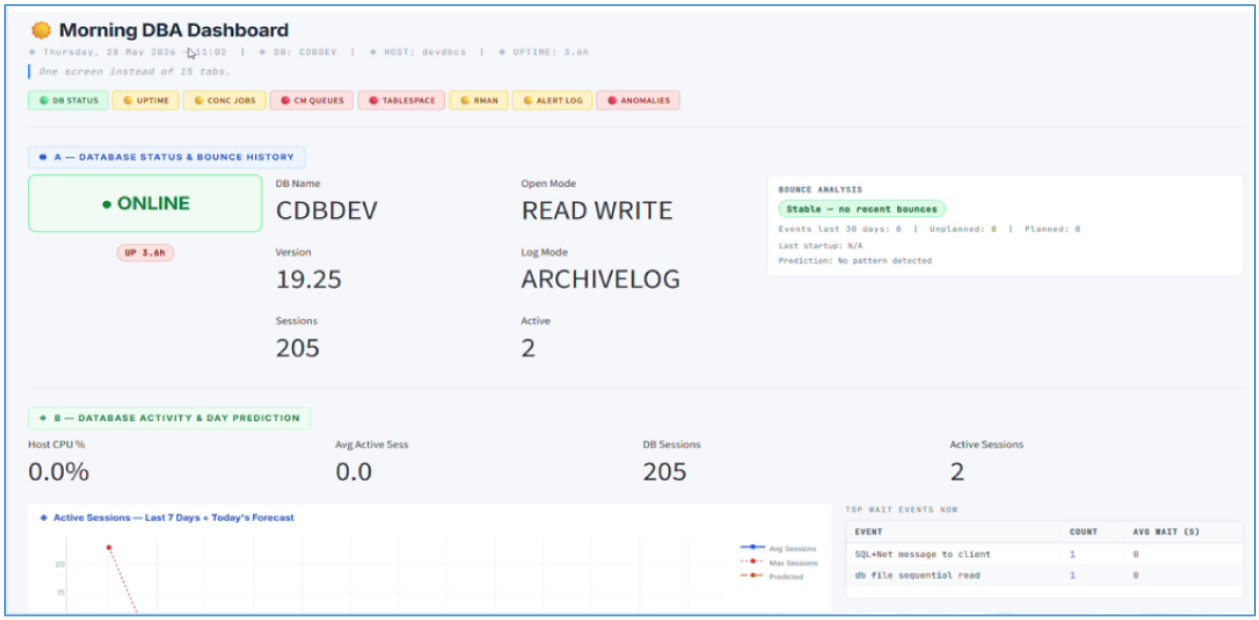
No machine learning in this script. The compute is pure logic: calculate the used percentage for each tablespace, apply traffic light rules (above 90% = red, above 75% = amber, otherwise green), rank schemas by size, and filter out built-in Oracle system schemas. The pandas library organises all query results like a spreadsheet in memory. Python making simple decisions — automatically, every time the page loads.

Communicate

Streamlit renders the entire page as a live web dashboard — no web development or HTML knowledge required. Plotly draws interactive bar charts showing used vs total space per PDB, side by side. Metric tiles show database name, version, and open status. Hover over any bar to see exact MB values. A DBA who knows Python can build this in an afternoon, and the result looks like something an IT team spent three months building. Every bar is your actual database, right now. Not a screenshot from last week.

10b. Demo 2 – Morning DBA Dashboard

The Morning DBA Dashboard is the morning briefing your DBA wishes they had every day. Ten sections covering database health, EBS concurrent jobs, disk usage, RMAN backups, alert log errors, anomaly detection, and AI-generated recommendations — all on a single page. The real story is what happens behind the scenes. This dashboard does not just report. It predicts. It detects. It recommends.



Collect

The script collects from 15+ Oracle data sources simultaneously. V$DATABASE and V$INSTANCE provide database identity and uptime. V$ACTIVE_SESSION_HISTORY delivers seven days of hourly session counts — the raw material for the session forecast. V$SYSTEM_EVENT captures wait events happening right now. V$SYSMETRIC provides live CPU usage directly from Oracle. V$DIAG_ALERT_EXT surfaces ORA- errors from the alert log without needing file system access. DBA_TABLESPACE_USAGE_METRICS, DBA_DATA_FILES, and DBA_SEGMENTS cover space across tablespaces, datafiles, and segments. V$RMAN_BACKUP_JOB_DETAILS provides every RMAN backup job for the last 14 days, and V$RECOVERY_FILE_DEST reports Flash Recovery Area usage. At the OS level, /proc/mounts combined with Python's shutil library reads real disk usage from every mounted filesystem. For EBS, FND_CONCURRENT_REQUESTS, FND_USER, and

FND_CONCURRENT_PROGRAMS_TL cover the full concurrent job picture. Imagine 15 analysts, each an expert in their area, handing you a report at the same time every morning. That is this script. It costs nothing and takes three seconds.

Connect

Two Oracle connections are established at startup. The first connects as SYS via SYSDBA mode for all V$ views, RMAN data, alert log, and disk information. The second connects as the APPS user for all EBS FND tables. The oracledb library's init_oracle_client function activates thick mode, enabling support for Oracle's Native Network Encryption required by most EBS environments. Two connections, two different worlds — the database engine and the EBS application — both working simultaneously through one Python script.

Compute – Where the Machine Learning Lives

Five ML and statistical techniques run inside this script, each solving a real DBA problem.

Session Forecasting (Linear Regression): Seven days of hourly session counts from V$ACTIVE_SESSION_HISTORY are used to train a LinearRegression model from scikit-learn. The model then predicts the next 8 hours of expected session load. This is a weather forecast for your database traffic — the model tells you at 2 PM today to expect 180 sessions. Your team plans capacity before the load hits, not after users start calling.

Database Size Forecast (Linear Regression): Thirty days of database size history from V$RMAN_BACKUP_JOB_DETAILS train the same LinearRegression model on growth trends. The model projects the database size 30 days into the future. No more midnight calls because a disk filled up. You knew three weeks ago.

Tablespace Countdown (Linear Regression): For each tablespace, the growth rate is calculated and projected forward to predict how many days remain before it reaches 100% capacity. Every tablespace now has a countdown clock. "SYSAUX — 14 days until full." That is not a panic alert. That is a planning tool.

RMAN Space Forecast (numpy): The average size of the last N RMAN backup jobs is calculated using numpy's np.mean function. That average is divided into the available FRA space to predict how many backup runs remain before the FRA is full. "You have space for five more RMAN backups." Before this script, a DBA found out the FRA was full when the backup failed at 3 AM. Now they find out on Tuesday afternoon.

Anomaly Detection (Z-Score): The statistical mean and standard deviation of hourly concurrent job counts, error rates, and session volumes are calculated using numpy. The Z-score formula — (value minus mean) divided by standard deviation — is applied to every data point. Any value more than two standard deviations from the mean is flagged as anomalous. Every Tuesday at 2 PM you normally run 50 concurrent jobs. This Tuesday you ran 180. Nobody called. No alert fired. But the Z-score model saw it and flagged it. The script memorises your database's normal heartbeat and immediately notices when it changes.

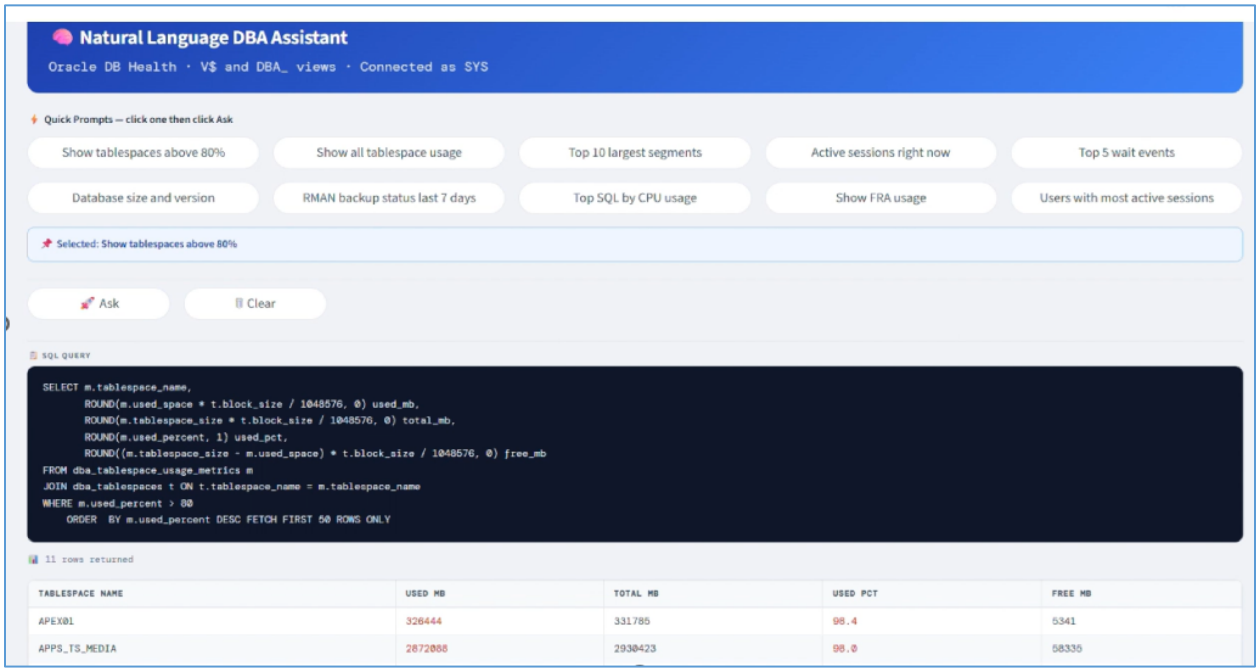
AI Recommendations (Rules Engine): All collected data feeds a prioritised rules engine that checks for critical conditions: database down, unplanned bounces, concurrent managers stopped, FRA nearly full, ORA-600 or ORA-7445 errors, and tablespaces above 90%. The output is a ranked action list — red for critical, yellow for warning. Your DBA's morning checklist, automated. The script goes through 10 different screens and hands you the three things you need to deal with today, in order, with reasons. It does not replace the DBA. It makes the DBA 10x faster.

Communicate

Ten colour-coded sections (A through J) cover every area of the database estate. A health strip at the top of the page shows eight green, amber, or red lights — instant status across eight dimensions at a single glance. Plotly charts display session history with an 8-hour forecast overlay and a 30-day database size trend. Forecast badges show projected DB size and RMAN runs remaining. RMAN job cards are green for success and red for failure, each showing duration and backup size. Pure HTML tables bypass Streamlit's rendering limitations to ensure all data is always visible. ORA- error cards are grouped by error code and sorted by frequency. The anomaly list highlights anything statistically unusual in plain bullet points. Everything the script knows is distilled into a page you can read in 90 seconds.

10c. Demo 3 – Natural Language DBA Assistant

The Natural Language DBA Assistant is the demo that makes audiences go quiet. Everything before it answered questions we already knew to ask. This one answers questions you have not thought of yet. You type a question in plain English. It talks to your Oracle database. And it brings the answer back — along with the SQL it wrote and a plain English summary of what it found.



Collect

Two modes cover two different data universes. DB Health mode queries Oracle V$ views and DBA_ views — the database engine. EBS Concurrent mode queries Oracle EBS FND tables — the application layer. Unlike the dashboards that pre-fetch everything at load time, this script collects nothing until you ask a question. Every prompt triggers a fresh live query against the database. Like a very efficient assistant who does not prepare 50 reports — they answer the one question you actually have right now.

Connect

This script makes two types of connection in sequence, which is what makes it unique. The first is to Oracle via oracledb — switching between SYS and APPS depending on the selected mode. The second is to the Anthropic Claude AI API via HTTPS — your question travels to Claude's servers and a response returns in approximately two seconds. The anthropic Python library handles this with a single function call: client.messages.create. A few years ago, connecting to an AI required a research team and millions of dollars. Today it is four lines of Python and a five-dollar API credit.

Compute

The compute happens in three stages. First, the question is sent to Claude along with a carefully engineered system prompt that instructs it to behave as an Oracle DBA expert, specifies the exact tables and views it is allowed to use, and enforces critical rules such as never joining FND_CONCURRENT_PROGRAMS_TL directly — a known Oracle EBS synonym issue that causes ORA-00904 errors. Claude returns only valid, executable SQL. Second, that SQL is executed against the live Oracle database and the results are processed by pandas into a DataFrame. Third, the results are sent back to Claude in a second API call, asking for a plain English summary. Claude reads the data and writes: "14 jobs failed in the last seven days. AP Invoice Import failed nine times — the most frequent failure." Two round trips to Claude. One to write the question. One to explain the answer. The compute is split between Anthropic's servers and your Oracle database, collaborating in about four seconds.

Communicate

Every question produces three outputs. A dark code box displays the generated SQL with syntax highlighting, formatted by the sqlparse library. A colour-coded results table shows the query output with red, amber, and green applied automatically to percentage columns. A blue summary card below the results contains Claude's plain English explanation — written for a manager, not a DBA. A mode badge in the sidebar shows whether you are in DB Health or EBS Concurrent mode, and a Test Connection button confirms the database is live before the demo begins. Three pieces of information. Every time. One click. The SQL for your technical audience. The results for everyone. The summary for your manager.

11. The Fusion SaaS Visibility Problem

Managing Oracle Fusion SaaS often means working with limited visibility into what is actually happening inside your own environment and depending entirely on Oracle Support for answers that should be at your fingertips.

> *Hands up who has ever felt like Fusion is basically a beautiful black box with a login screen? Yeah. Same hands. Every time.*
>
> *Something goes slow. A user calls you. You log in. You click around. You open the Scheduled Processes screen. You stare at it. You have absolutely no idea what's happening inside until Oracle Support tells you three business days later. That's not monitoring. That's hoping.*

Every SR for a Fusion performance issue follows the same script: "Please provide the request ID." "Please provide the date and time in UTC." "Please clear your cache and try again." Meanwhile your CFO is standing behind you asking why month-end close is running slow.

The OIC monitoring challenge is equally frustrating:

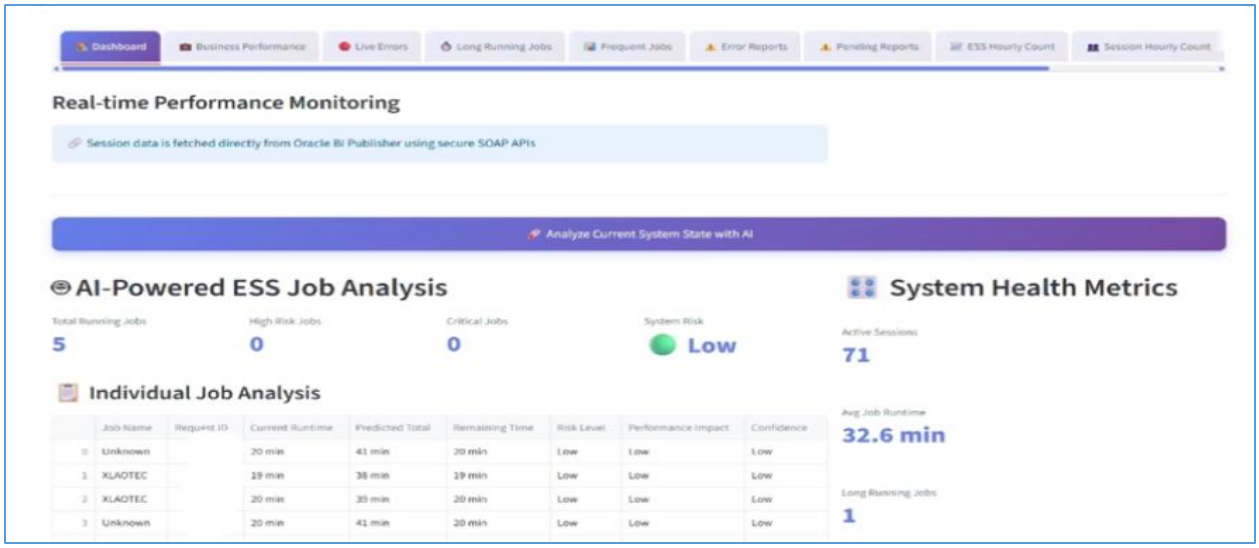
- Monitoring data disappears after 10 days sometimes less.
- No historical trending or long-term pattern analysis.
- No visibility into why an integration failed two weeks ago.
- ESS jobs return only success or failure no predictive insight.

So to summarise the Fusion SaaS monitoring experience: you discover issues right after the business does. Your historical data has an expiry date. Your visibility depends on Oracle Support response time. And your monitoring strategy is fifteen tabs and optimism.

"This is exactly the problem we're going to solve with Python, ML, and a sidekick that never forgets and never asks you to clear your cache."

11a. Demo 4 – Oracle Fusion SaaS Dashboard

The Fusion SaaS Dashboard moves the sidekick from the database layer to the application layer. Fusion is a cloud application — you cannot query it like a local database. It has its own APIs, its own reporting engine, its own security model. Python has to speak Fusion's language. Ten tabs. Four ML techniques. An Azure OpenAI integration. And an automated alerting engine that watches the dashboard even when you are not looking at it.



Collect & Connect – Three Data Sources

Three completely different data sources feed this dashboard, each using a different protocol and Python library.

Fusion REST API: Python uses the requests library with HTTP Basic Auth or OAuth to call Fusion's REST endpoints. This collects ESS job history, job status, running and long-running jobs, and job parameters. Python is knocking on Fusion's front door and asking what jobs ran today and which ones failed.

Oracle BI Publisher via SOAP: The zeep Python library calls BIP reports over SOAP — the same way Fusion's own user interface does internally. The script sends a SOAP XML request, BIP executes the report, returns base64-encoded data, Python decodes it, and pandas converts it to a DataFrame. This collects session data, business performance reports, and detailed job execution information. Python is running Fusion's own reports automatically, without ever needing to log in.

Oracle ATP – Autonomous Transaction Processing: All data collected from REST and BIP is pushed into Oracle ATP — Oracle's managed cloud database in OCI. The oracledb library connects to ATP using Oracle Wallet authentication, providing certificate-based encryption end to end. ATP becomes the single source of truth and historical store. The dashboard then reads from ATP to display everything. REST and BIP collect. ATP stores. Dashboard displays. That is the clean pipeline. All stored passwords are protected using Fernet symmetric encryption from the cryptography library, so configuration files are safe even if accessed by unauthorised parties.

Compute – Four ML Techniques

Isolation Forest – Anomaly Detection: sklearn's IsolationForest monitors ESS job runtimes and database session behaviour. It learns what normal looks like from historical data without requiring any manual rules or labelled training data. If a job that normally completes in two minutes suddenly takes 45, Isolation Forest flags it before your users do. Think of it as a security guard who studies everyone's routine for a week, then immediately notices when something is out of place.

Random Forest Classifier – Risk Scoring: sklearn's RandomForestClassifier builds 100 decision trees, each trained on historical ESS job data: runtime patterns, error frequency, time of day, and submitting user. Each tree votes on the risk level — High, Medium, or Low — and the majority wins. The output is not just "this job failed" but "this job has a 78 percent probability of being a problem based on everything we know about it." That is the difference between a log and intelligence.

Ensemble Capacity Forecast – Triple Model: Three models run simultaneously and their results are weighted and combined: Linear Regression at 30 percent, Polynomial Regression degree 2 at 30 percent, and Random Forest Regressor at 40 percent. This ensemble predicts future ESS request volumes for up to 90 days and displays confidence bands — the upper and lower bounds of the prediction — built from the variance across all 100 Random Forest trees. Three forecasters, each seeing the data differently, weighted by reliability, with an honest range showing where the future is likely to land. "At current growth rate, you will hit capacity in X days."

Azure OpenAI GPT-3.5 – AI Narrative: After anomaly detection and risk scoring complete, the results are sent to Azure-hosted GPT-3.5-turbo via the openai.AzureOpenAI client. GPT writes a plain English narrative: what was found and what to do about it. Because this uses the Azure-hosted version of GPT rather than the public OpenAI API, the data stays within your

organisation's Azure tenant and never leaves your enterprise security boundary. The ML finds the problems. The AI explains them. One sentence your manager can read.

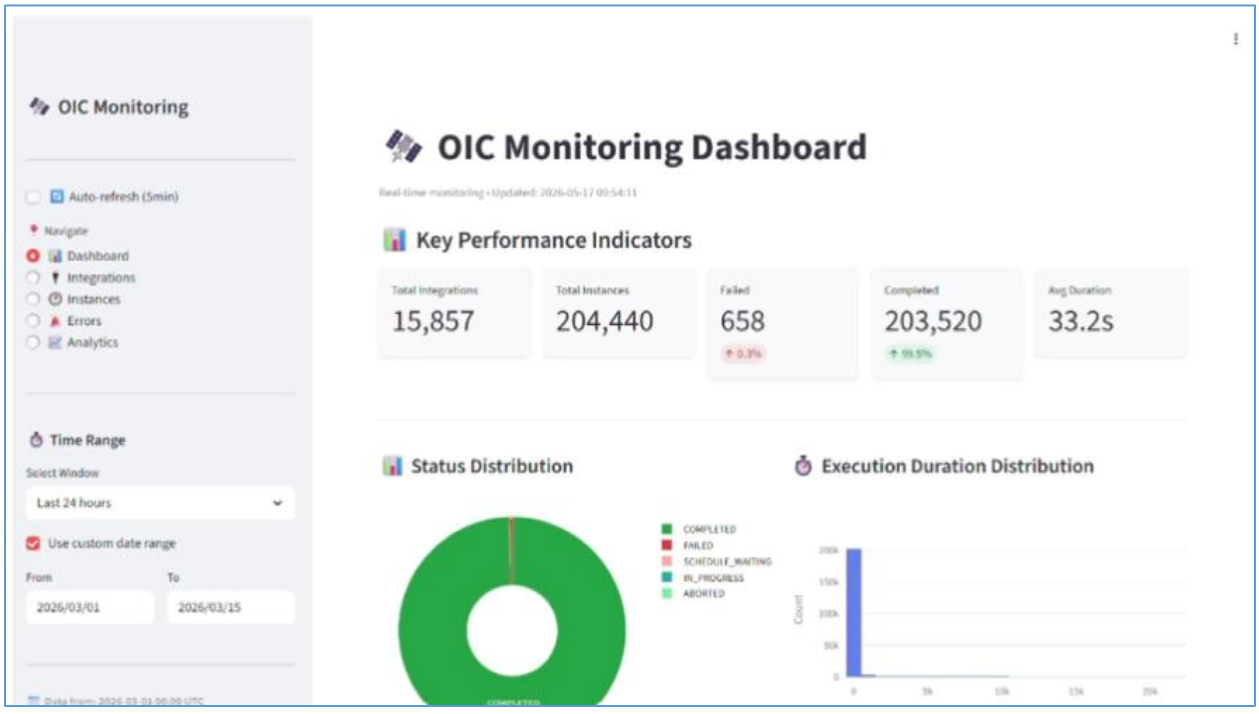
Automated Alerting (APScheduler + smtplib): A BackgroundScheduler from the APScheduler library runs silently while the dashboard is open, checking resource pressure and performance thresholds every few minutes. When a critical limit is breached, smtplib sends an email alert to your team with cooldown logic to prevent alert storms. The dashboard does not wait for you to look at it. It looks at it for you.

Communicate

Ten tabs deliver complete visibility: Dashboard (health metrics and AI analysis), Business Performance (BIP report data and KPIs), Live Errors (real-time categorised failures), Long Running Jobs (jobs exceeding thresholds), Frequent Jobs (most-run programs with parameter details), Error Reports, Pending Reports, ESS Hourly Count, Session Hourly Count, and Capacity Planning with ML forecast charts and confidence bands. Streamlit handles the tab layout. Plotly draws all interactive charts. Matplotlib and seaborn handle session analysis visualisations. The pytz library ensures all timestamps display in the configured local timezone. All stored passwords are encrypted using Fernet, and the entire dashboard reads its data from ATP — one consistent source of truth across all ten tabs.

11b. Demo 5 – OIC Monitoring Dashboard

Oracle Integration Cloud is the middleware — the plumbing connecting Fusion to everything else: payroll providers, third-party APIs, legacy applications. When OIC works, nobody knows it exists. When OIC breaks, everyone knows immediately. This dashboard makes sure you know first, with a complete data pipeline from OIC's REST API all the way through to a live visual dashboard via ATP.



Collect & Connect – The Four-Step Pipeline

A dedicated Python script — OIC_to_atp.py — runs on a schedule and feeds the dashboard. It is the data pipeline, and it works in four steps.

Step 1 – OAuth Token: Python calls Oracle IDCS (Identity Cloud Service) with a client ID and client secret and receives a Bearer token in return. This token expires automatically after a set period. No username or password travels across the network. Modern, secure, fully automated.

Step 2 – OIC REST API: Using the Bearer token, the requests library makes paginated calls to OIC's monitoring REST endpoints. Everything is collected: the full integrations catalogue, every execution instance, all error records with codes and messages, error item detail, and execution tracking data.

Step 3 – Push to ATP (MERGE INTO): Everything fetched from OIC is written to Oracle ATP using MERGE INTO — a smart upsert that inserts new records and updates existing ones with no duplicates. Four ATP tables store the full history: OIC_INTEGRATIONS_HISTORY, OIC_MONITORING_INSTANCES_HIST, OIC_MONITORING_ERRORS_HIST, and OIC_MONITORING_ERROR_ITEMS_HIST. The oracledb library connects to ATP with Oracle Wallet, providing certificate-based encryption end to end. OIC's native monitoring data disappears after 10 days. ATP keeps it indefinitely for trend analysis.

Step 4 – Dashboard Reads from ATP: The OIC dashboard queries the ATP tables directly. Every chart, every table, every metric reads from ATP. A time window selector in the sidebar — Last 1 Hour, 6 Hours, 24 Hours, 7 Days, or 30 Days — updates every single query across the entire dashboard simultaneously. OIC API collects. Python processes. ATP stores. Dashboard displays. One script to collect, one script to display.

Compute

No machine learning in this script — and that is intentional. The data tells the story without needing models. SQL-driven intelligence calculates success rate per integration, duration trends (average, minimum, and maximum execution times), and error frequency grouped by error code. Automatic status badges map COMPLETED to green, FAILED to red, and RUNNING to amber. An optional five-minute auto-refresh keeps the dashboard live with zero manual interaction — ideal for watching a period-end close or a go-live. Three numbers on the Errors page — total errors, affected instances, unique error codes — give your entire integration health story in a single glance. The difference between a four-hour investigation and a ten-minute fix.

Communicate

Five pages are accessible from a sidebar radio navigation. The Dashboard page shows headline metrics, a status distribution pie chart, and the slowest integrations table. The Integrations page provides a searchable and filterable catalogue of all integrations, allowing drill-down into any integration's full execution history. The Instances page shows every individual execution record, filterable by integration name, status, and sort order. The Errors page delivers root cause analysis: top error codes with occurrence counts, affected instances, and full error message detail with drill-down to error items and links. The Analytics page shows success rate trends over time, average duration trends, and integration performance rankings. Streamlit handles sidebar navigation and layout. Plotly draws all charts — pie, line, bar, and histogram. The cryptography.Fernet library encrypts all stored credentials. The pytz library ensures all timestamps display in local time.

12. The New Beginning Tools Evolve, Humans Adapt

Every generation of tools was once considered revolutionary and every generation prompted the same fear: "Will this replace us?" History answers that question clearly every time.

From stone carving tools to feather quills, from fountain pens to keyboards, and now to AI assistants each transition upgraded human capability, it did not eliminate it.

> *The printing press was going to replace scribes. The typewriter was going to replace writers. The computer was going to replace everyone. And yet here we are. More connected, more productive, and more employed than ever.*
>
> *Each tool required something different. The stone tool required strength. The quill required patience. The keyboard required skill. The AI assistant requires curiosity. That's it. Just the willingness to learn something new.*

"The AI assistant requires only one thing: curiosity. And if you're in this room today, you already have it."

---

13. The Fifth C Collaborate

And finally, the reveal. After Collect, Connect, Compute, and Communicate the fifth C is the most important one of all.

C5 = COLLABORATE

Not Automate. Not Replace. Not Eliminate. Collaborate.

AI is not your replacement. It is your colleague a very fast, very tireless, never-complaining colleague who also does not steal your lunch from the office fridge.

Here is what collaboration with AI looks like in practice:

- AI highlights critical alerts you focus on the fix.
- AI summarises logs you make the decision.
- AI predicts issues you prevent them.
- AI generates documentation, you take the credit.

The expertise, the judgment, the context, the experience that is still you. AI removes the noise so the real value of what you do gets to shine.

> *And to all my fellow DBAs and consultants the ones who've been woken up at 2 AM, the ones who've scrolled through alert logs on Christmas morning, the ones who've been asked to "just quickly check something" on a Friday afternoon:*

"AI won't replace you. But it might finally let you sleep through the night."

Keep calm. Carry on. And go build your sidekick.

14.The Bonus Part

> ⚠ **Please Read Before Running Anything**
>
> **The scripts and instructions in this github link are provided as-is, for educational and demonstration purposes only.**
>
> **By downloading, modifying, or running any script from this session, you agree to the following:**
>
> **The good news first:** Every single query in these scripts is a SELECT statement. No INSERT. No UPDATE. No DELETE. No DROP. No ALTER. Nothing is written to your database. Nothing is changed. Nothing is deleted. Your data is completely safe. Your EBS instance will not know these scripts exist.
>
> **The fine print:** While every care has been taken to ensure these scripts work correctly, **Akhash Ramnath** and **Doyensys** accept no responsibility for any issues that arise from running these scripts in your environment — including but not limited to: unexpected errors, server load, configuration conflicts, or your manager walking in while the screen shows 96% disk usage and asking uncomfortable questions.
>
> **In plain English:** These scripts work perfectly in the environments they were built and tested on. Your environment may be different. Test in a non-production environment first. Read the code before you run it. You are a DBA — you know better than anyone that you should never run something you do not understand.
>
> **Run at your own risk. We are not responsible. But also relax. It is just a SELECT.**
>
> ---
>
> ***"The only thing these scripts can do to your database is look at it. Which, ironically, is more than most monitoring tools do."***

Click here to download:

https://github.com/akhashramnath/DIY-Build_Your_Own_AI_Sidekick_Oracle

Connect with the Author

Akhash Ramnath
Sr. Principal Consultant | Oracle ACE Professional
akhash.ramnath@doyensys.com