Empowering Database Administration with AI

A New Era of Insight & Efficiency

Akhash Ramnath

DOYENSYS
VALUE INNOVATION

About this Article

This brief write-up focusses on the intersection of Artificial Intelligence (AI) and Machine Learning (ML) is revolutionizing in the field of database administration.

Leveraging the AI and ML technologies, DBA's could automate routine tasks, gain deeper insights into their data, and enhance overall database performance and security.

We will initially explore how the role of a database administrator had changed over the period of time and how the AI and ML works.

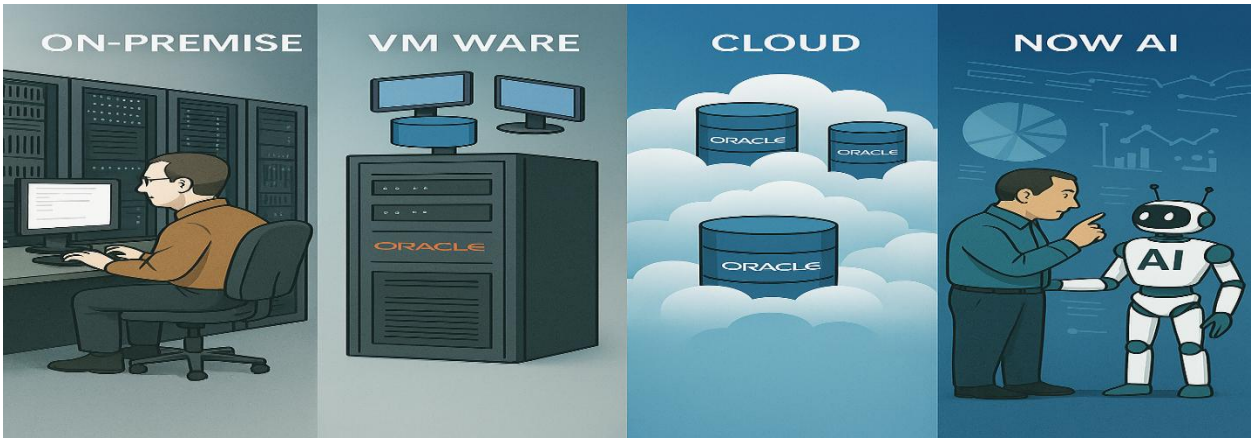
Later this article highlights few of the features developed with AI and ML to provision the daily DBA activities. Although my background is primarily in Oracle, the use cases presented will be applicable across various database platforms.

This read is focused on sharing some of the best practices to be commemorated in your environment for a seamless and productive Database habitat.

Evolution of Database Administration: Enter AI

A Database Administrator (DBA) is responsible for building, managing, maintaining updating/upgrading the database systems, while also ensuring the performance, integrity and security of the data.

The evolution of database management has been marked by significant shifts in technology and infrastructure.



This evolution from on-premises to AI has been driven by a desire for greater flexibility, scalability, cost-efficiency, and automation. While the evolution of database management from on-premises to AI has brought many benefits, there are also some potential challenges to consider:

Data Governance

Data Growth: Data growth has significantly increased in modern database systems due to the Digital Transformation, E-commerce, Research & Social Media, Internet Of Things, etc.

Data Quality: Tracking the lineage of data and ensuring the data quality and consistency can be complex, especially in hybrid and multi-cloud setups.

Increased Complexity

Hybrid Environments: The transition to hybrid environments, combining on-premises and cloud resources, can introduce additional complexity in terms of management and coordination.

Security Concerns

Data Privacy: Ensuring data privacy and compliance with regulations becomes more challenging in complex, distributed environments.

Data Breaches: The increased reliance on cloud infrastructure and external services can expose organizations to new security risks.

AI and ML can be a powerful solution to handle all these challenges and by carefully planning and addressing these potential issues appropriately.

How AI Works

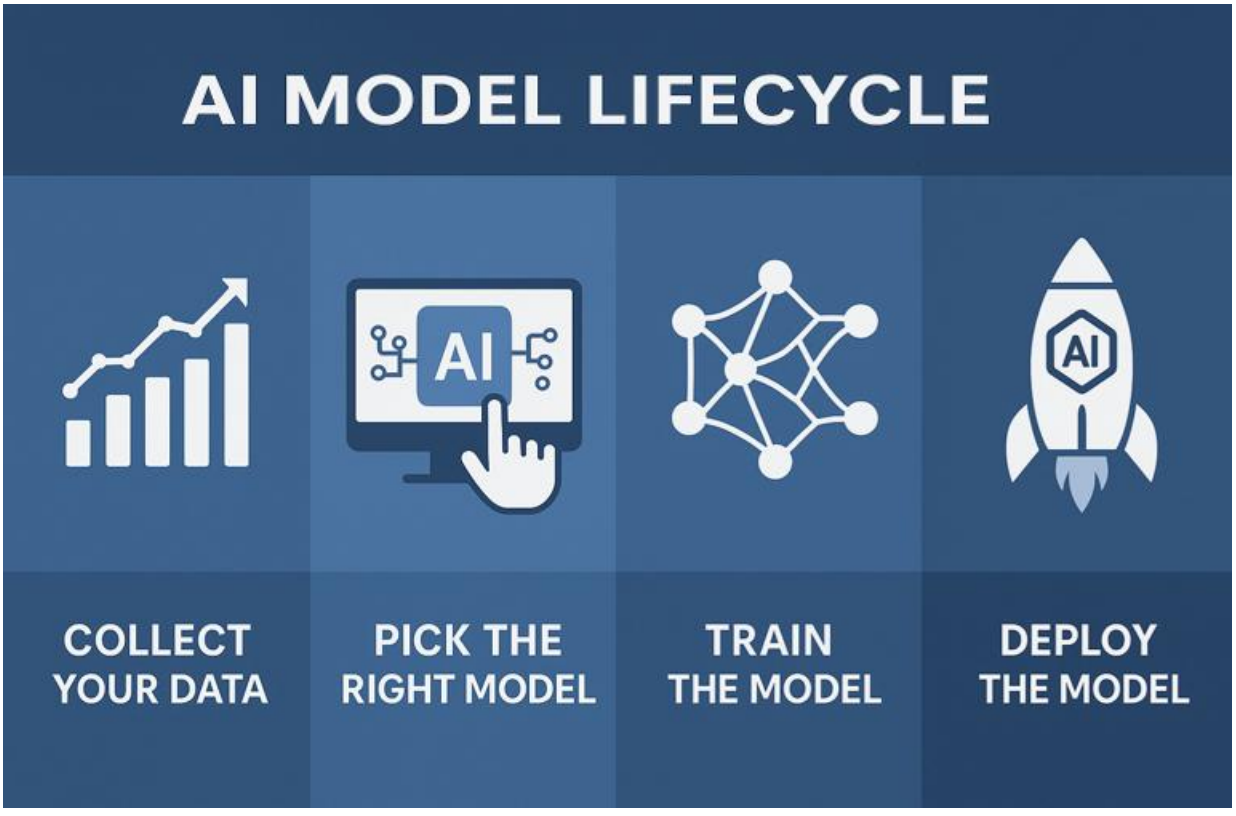
***AI** is simply a system that performs tasks that typically require human intelligence, such as:*

Understanding and responding to natural language; Recognizing patterns and making predictions; Solving problems & Learning from experience

***Machine Learning** is a specific subset of AI that focuses on developing algorithms that allow computers to learn from data. These algorithms can identify patterns, make predictions, and improve their performance over time*

***AI Models:** These are the specific algorithms and their trained parameters that are used to make predictions or decisions.*

***ML Libraries:** These are collections of pre-built algorithms and functions that make it easier for developers to implement ML models.*



The perfect way to invoke AI in database administration is to build a perfect AI Model that supports your ecosystem.

AI in Database Administration

The Database Administrators could very well leverage the AI to support them with:

- *Automating Tasks.*
- *Enhancing System Performance.*
- *Troubleshooting and identifying the root cause.*
- *Security and Data Monitoring.*
- *Documentation tasks.*

If you are not sure how or where to start!!! This is the best way to step into AI.

Start Simple!!!

The Best way to get used to AI is to introduce python and machine learning scripts in to your database and application. Replace all your regular shell scripts to Python scripts.

This is my first Python Script "Hello World"

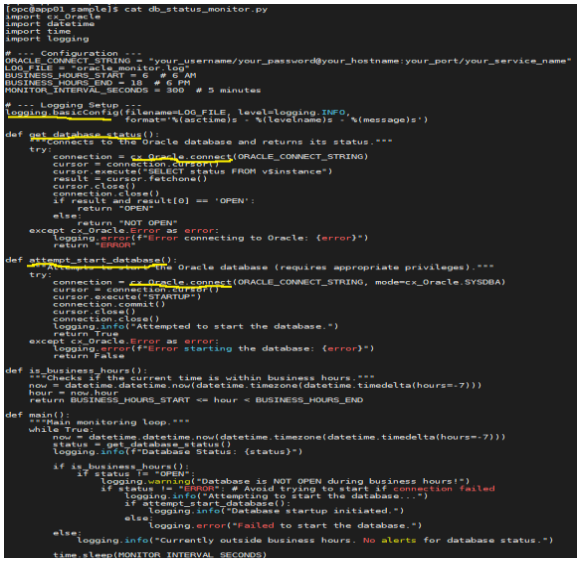
```
[opc@app01 sample]$
[opc@app01 sample]$ python --version
Python 2.7.5
[opc@app01 sample]$ python hello_world.py
Hello, World! This is running on the Oracle DB server.
[opc@app01 sample]$
```

Install the necessary libraries cx_oracle, pandas, numpy, scikit-learn, etc

```
opc@app01:~$ pip install cx_Oracle pandas numpy scikit-learn matplotlib seaborn streamlit
Collecting cx_Oracle
  Downloading cx_Oracle-8.3.0-cp39-cp39-manylinux1_x86_64.whl (800 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 800.3/800.3 kB 3.2 MB/s eta 0:00:00
Collecting pandas
  Downloading pandas-2.2.2-cp39-cp39-manylinux_2_17_x86_64.whl (12.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.4/12.4 MB 6.8 MB/s eta 0:00:00
Collecting numpy
  Downloading numpy-1.26.4-cp39-cp39-manylinux_2_17_x86_64.whl (15.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.9/15.9 MB 9.1 MB/s eta 0:00:00
Collecting scikit-learn
  Downloading scikit_learn-1.4.1.post1-cp39-cp39-manylinux_2_17_x86_64.whl (9.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.5/9.5 MB 7.8 MB/s eta 0:00:00
Collecting matplotlib
  Downloading matplotlib-3.8.4-cp39-cp39-manylinux_2_17_x86_64.whl (8.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.5/8.5 MB 5.6 MB/s eta 0:00:00
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl (302 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 302.1/302.1 kB 2.7 MB/s eta 0:00:00
Collecting streamlit
  Downloading streamlit-1.32.2-py2.py3-none-any.whl (11.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.0/11.0 MB 6.2 MB/s eta 0:00:00
```

Few basic python scripts

Database Status Check



```
[opc@app01 sample]$ cat db_status_monitor.py
import cx_Oracle
import datetime
import time
import logging

# --- Configuration ---
ORACLE_CONNECT_STRING = "your_username/your_password@your_hostname:your_port/your_service_name"
LOG_FILE = "oracle_monitor.log"
BUSINESS_HOURS_START = 6  # 6 AM
BUSINESS_HOURS_END = 18  # 6 PM
MONITOR_INTERVAL_SECONDS = 300  # 5 minutes

# --- Logging Setup ---
logging.basicConfig(filename=LOG_FILE, level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

def get_database_status():
    """Connects to the Oracle database and returns its status."""
    try:
        connection = cx_Oracle.connect(ORACLE_CONNECT_STRING)
        cursor = connection.cursor()
        cursor.execute("SELECT status FROM v$instance")
        result = cursor.fetchone()
        cursor.close()
        connection.close()
        if result and result[0] == 'OPEN':
            return "OPEN"
        else:
            return "NOT OPEN"
    except cx_Oracle.Error as error:
        logging.error(f"Error connecting to Oracle: {error}")
        return "ERROR"

def attempt_start_database():
    """Attempts to start the Oracle database (requires appropriate privileges)."""
    try:
        connection = cx_Oracle.connect(ORACLE_CONNECT_STRING, mode=cx_Oracle.SYSDBA)
        cursor = connection.cursor()
        cursor.execute("STARTUP")
        connection.commit()
        cursor.close()
        connection.close()
        logging.info("Attempted to start the database.")
        return True
    except cx_Oracle.Error as error:
        logging.error(f"Error starting the database: {error}")
        return False

def is_business_hours():
    """Checks if the current time is within business hours."""
    now = datetime.datetime.now(datetime.timezone(datetime.timedelta(hours=-7)))
    hour = now.hour
    return BUSINESS_HOURS_START <= hour < BUSINESS_HOURS_END

def main():
    """Main monitoring loop."""
    while True:
        now = datetime.datetime.now(datetime.timezone(datetime.timedelta(hours=-7)))
        status = get_database_status()
        logging.info(f"Database Status: {status}")

        if is_business_hours():
            if status != "OPEN":
                logging.warning("Database is NOT OPEN during business hours!")
                if status != "ERROR": # Avoid trying to start if connection failed
                    logging.info("Attempting to start the database...")
                    if attempt_start_database():
                        logging.info("Database startup initiated.")
                    else:
                        logging.error("Failed to start the database.")
        else:
            logging.info("Currently outside business hours. No alerts for database status.")

        time.sleep(MONITOR_INTERVAL_SECONDS)
```

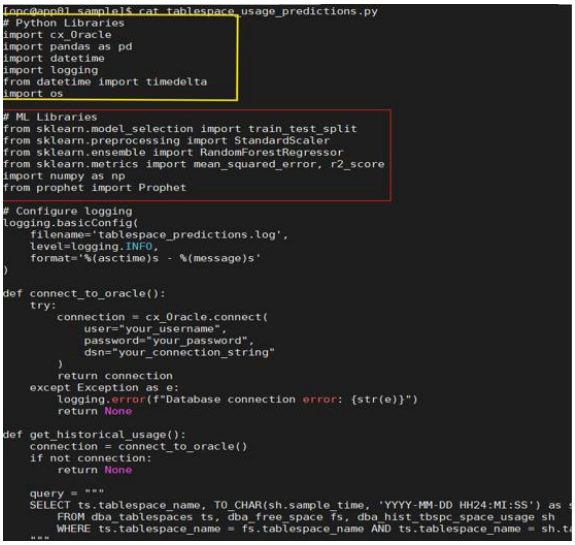
This is a simple database monitoring python script that alerts the DBA's if the database status is unavailable and also maintains a log history of the database availability.

No ML Libraries used.

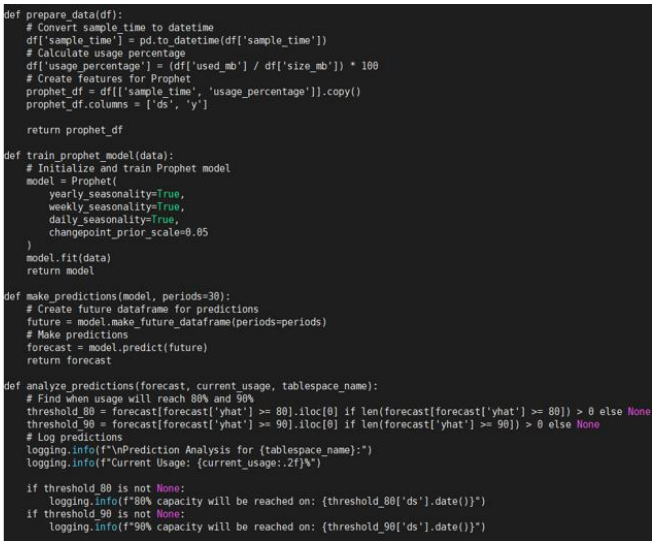
Basic python modules/libraries like cx_oracle, datetime, time, logging are used

Tablespace Monitoring: Predict Tablespace Usage

This is another typical Tablespace monitoring python script, not only it alerts the DBA's in case the database tablespace's are running out of space it also predicts the tablespace growth. Yes!!! This python script can predict how much a tablespace might grow in size in the near future. This would help the DBA's to make necessary storage planning way in advance and could totally avoid any inconveniences caused due to the tablespace full errors.



```
topc@app01 sample1$ cat tablespace_usage_predictions.py
# Python Libraries
import cx_Oracle
import pandas as pd
import datetime
import logging
from datetime import timedelta
import os

# ML Libraries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
from prophet import Prophet

# Configure logging
logging.basicConfig(
    filename='tablespace_predictions.log',
    level=logging.INFO,
    format='%(asctime)s - %(message)s'
)

def connect_to_oracle():
    try:
        connection = cx_Oracle.connect(
            user="your_username",
            password="your_password",
            dsn="your_connection_string"
        )
        return connection
    except Exception as e:
        logging.error(f"Database connection error: {str(e)}")
        return None

def get_historical_usage():
    connection = connect_to_oracle()
    if not connection:
        return None

    query = """
    SELECT ts.tablespace_name, TO_CHAR(sh.sample_time, 'YYYY-MM-DD HH24:MI:SS') as s
        FROM dba_tablespaces ts, dba_free_space fs, dba_hist_tbspc_space_usage sh
        WHERE ts.tablespace_name = fs.tablespace_name AND ts.tablespace_name = sh.t
    """
```



```
def prepare_data(df):
    # Convert sample_time to datetime
    df['sample_time'] = pd.to_datetime(df['sample_time'])
    # Calculate usage percentage
    df['usage_percentage'] = (df['used_mb'] / df['size_mb']) * 100
    # Create features for Prophet
    prophet_df = df[['sample_time', 'usage_percentage']].copy()
    prophet_df.columns = ['ds', 'y']

    return prophet_df

def train_prophet_model(data):
    # Initialize and train Prophet model
    model = Prophet(
        yearly_seasonality=True,
        weekly_seasonality=True,
        daily_seasonality=True,
        changepoint_prior_scale=0.05
    )
    model.fit(data)
    return model

def make_predictions(model, periods=30):
    # Create future dataframe for predictions
    future = model.make_future_dataframe(periods=periods)
    # Make predictions
    forecast = model.predict(future)
    return forecast

def analyze_predictions(forecast, current_usage, tablespace_name):
    # Find when usage will reach 80% and 90%
    threshold_80 = forecast[forecast['yhat'] >= 80].iloc[0] if len(forecast[forecast['yhat'] >= 80]) > 0 else None
    threshold_90 = forecast[forecast['yhat'] >= 90].iloc[0] if len(forecast['yhat'] >= 90]) > 0 else None
    # Log predictions
    logging.info(f"\nPrediction Analysis for {tablespace_name}:")
    logging.info(f"Current Usage: {current_usage:.2f}%")

    if threshold_80 is not None:
        logging.info(f"80% capacity will be reached on: {threshold_80['ds'].date()}")
    if threshold_90 is not None:
        logging.info(f"90% capacity will be reached on: {threshold_90['ds'].date()}")
```

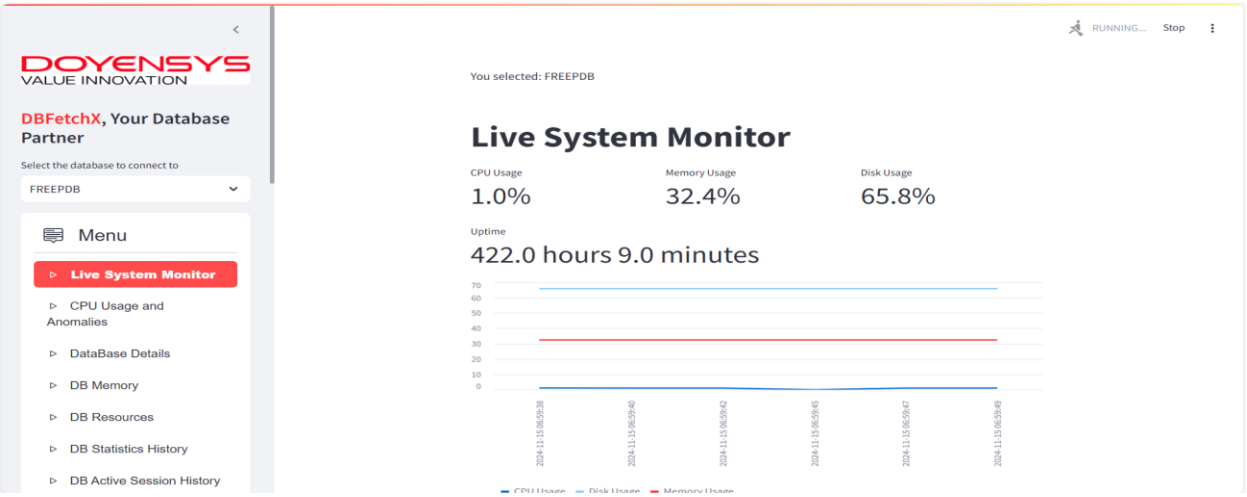
This is achieved by using well known ML libraries sklearn.model selection, sklearn.ensemble, etc.

Take it to Next Level: DBFetchX

Once you are comfortable with converting all the shell scripts to python scripts, you shall install libraries like pandas, numpy, scikit-learn, matplotlib, seaborn streamlit. Install and configure Oracle Instant Client. Connect to the database using cx_oracle and query db views/table. Use Pandas to organize the data, apply ML techniques for anomaly detection, trend forecasting, pattern recognition, etc and create a real time dashboard using Streamlit you shall have your own AI tool.

DBFetchX is an application built by Doyensys DBA team using Streamlit, Python, ML Libraries, SQL/PL-SQL .

We used some of the more popular reusable components posted in our company's blog and came up with this tool to simplify, streamline some of the common DBA activities like monitoring, performance tuning, capacity forecast, etc.

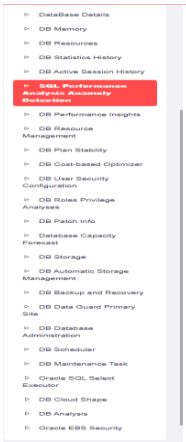


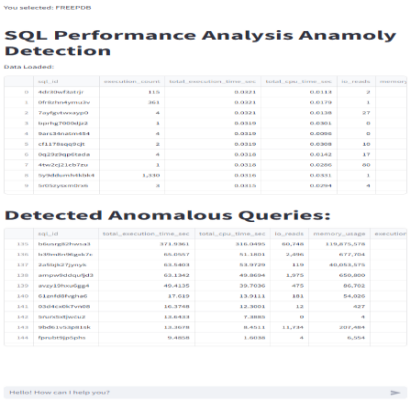
We will look into few examples where AI and ML had helped out the DBA's in their daily tasks.

Use Case 1: SQL Performance Anomaly

In this example, we have a script that monitors the database's performance and detects anomalies and alerts the DBA's in case of any CPU bottlenecks.

The script primarily gathers information regarding session data, SQL queries, CPU time, memory usage, I/O reads, and execution details from the database tables. It then analyses the collected data to forecast potential CPU or performance-related issues.

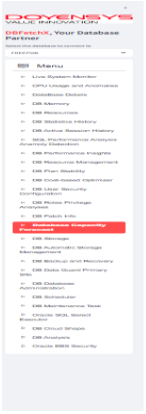




We used the simple sklearn.ensemble.Isolation module in our python scripts to detect anomalies.

Use Case 2: Database Capacity Forecast

This feature in the DBFetchX tool is designed to forecast the capacity management of the database. It tracks the growth of the database by gathering transaction information occurring within it and forecasts the potential future growth of the database.



It provides regarding the object, schema, tablespace, and datafile that have experienced significant alterations in size over a specified timeframe.

Use Case 3: CPU Usage Anomaly

This functionality the results from the "sar" command executed on the server, evaluates the data, and predicts future CPU usage. It alerts Database Administrators in the event of any detected anomalies that may suggest an impending performance bottleneck.

AI is your Side Kick

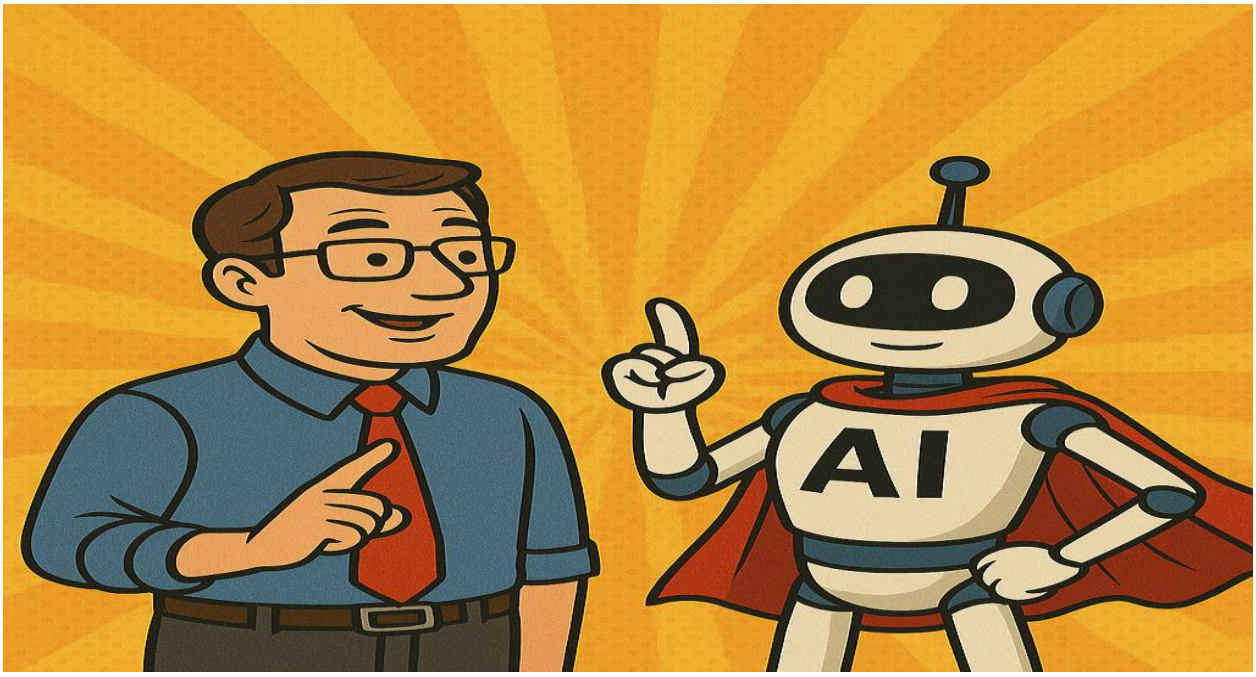
Employ AI as your Side Kick and delegate all your regular mundane task to AI.

- *AI Powered Task Automation: Automate the Mundane.*
- *AI-Driven Performance & Efficiency.*
- *Proactive Data Protection via AI.*
- *Effortless Data Management Using AI.*

By leveraging AI, DBAs can become more efficient, effective, and proactive in managing their databases.

Freeing up Us as DBAs to focus on strategic, high-value tasks, proactive initiatives and the future technologies.

Future Outlook: The Evolution of Database Administration with AI



Emerging trends in AI and database management,

- *Predictive and Self-Healing Databases, example: Oracle Autonomous.*
- *Data-Driven Automation and Decision Support.*
- *Query Generation and Natural Language Interface (NLI) for DB Management, example: Oracle APEX 24.*
- *Oracle AI Agents in Fusion, EBS and OCI Console.*
- *AI Prompts in Oracle Support.*

How AI Will Continue to Transform the DBA Role

- *Increased Focus on Strategy and Planning.*
- *A Shift Toward Proactive Database Health Monitoring.*
- *Enhanced Collaboration with AI Tools and Business Users.*
- *DBAs as 'Architects' and 'Strategists'.*

Conclusion:

To wrap up, we've seen how AI is transforming the role of Database Administrators—automating routine tasks, enhancing performance tuning, and enabling proactive issue resolution. By leveraging AI-driven tools, DBAs can shift from manual maintenance to strategic decision-making, unlocking greater efficiency and innovation.

However, successful adoption requires balancing AI's capabilities with human expertise, ensuring security, and fostering continuous learning. The future of database management isn't about replacing DBAs but empowering them to achieve more.

About Me:

I am an Oracle aficionado with a successful track record in the area of Oracle Cloud Fusion, Oracle E-Business Suite & Database Administration. I am also an Oracle Ace Associate with 13+ years of experience in implementation, support, and maintenance. I manage different areas of Oracle Databases like Real Application Cluster, Primary/Standby database||Dataguard, Oracle Golden Gate, Oracle E-Business Suite (11i, R12(R12.1/R12.2)), Oracle Cloud Fusion and Oracle Cloud Infrastructure. I am currently working in a reputed organization called Doyensys handling a demanding production environment, managing large volumes of business-critical data. I am a self-motivated, committed team player with the ability to communicate at all levels, he actively participates in the Oracle community by publishing technical posts on his blog at dbarepository.wordpress.com.

Connect with us:

Elevate your Database Ecosystem with Doyensys… Look upon how Doyensys provides productive and effective solution to keep your Oracle Journey functioning at peak operating efficiency.

Call +1-972-992-4220 or Visit : doyensys.com