Agentic Business Analytics

How AI Agents Change Data Work, Decision Support, and the Role of Analysts

Armando Vieira
Tartu University

May 2026

Abstract

Business analytics has traditionally been organized around dashboards, reports, notebooks, and scheduled pipelines. These remain important, but they do not fully match the way managers ask questions: "Why did this happen?", "What should I do next?", and "Which evidence supports the recommendation?" Agentic analytics adds a new layer. An AI agent can interpret a business question, decide which tools to use, query databases, retrieve internal documentation, reason across evidence, and produce a decision-oriented answer. This booklet explains the idea, shows where it fits in the analytics stack, and gives a detailed example of building an agent that accesses a SQL database and company documentation through retrieval-augmented generation (RAG).

Contents

1 From Business Analytics to Agentic Analytics

Business analytics is the practice of using data, statistical methods, machine learning, visualization, and business knowledge to improve decisions. In a conventional workflow, an analyst receives a question, writes SQL, checks dashboards, opens spreadsheets, maybe runs a model, and then writes a short interpretation.

That workflow is powerful, but it is often slow and fragmented. The evidence may be spread across:

- operational databases,
- BI dashboards,
- data warehouses,
- spreadsheets,
- product documentation,
- pricing policies,
- sales notes,
- marketing calendars,
- internal procedures and compliance rules.

Agentic business analytics changes the interface and the workflow. Instead of giving the user only a static dashboard, we can provide an assistant that can decide which analytical step is needed next. The agent does not replace the data platform. It sits on top of it as an orchestration and reasoning layer.

1.1 A useful definition

An **AI agent** is a system that combines:

- a language model,
- instructions that define its role and boundaries,
- tools it can call,
- memory or session state,
- guardrails and evaluation rules,
- an execution loop that lets it observe tool results and continue working.

In the OpenAI Agents SDK, the central mental model is:

$$\text{Agent} = \text{LLM} + \text{role} + \text{tools} + \text{workflow logic}.$$

The official Agents SDK documentation describes agents as systems run by a `Runner`; during a run, the model may produce a final answer, call tools, or hand off work to another agent. Function tools are ordinary Python functions wrapped so that the agent can call them when useful.[1]

[1] OpenAI Agents SDK documentation: https://openai.github.io/openai-agents-python/. Tools guide: https://openai.github.io/openai-agents-python/tools/. Running agents guide: https://openai.github.io/openai-agents-python/running_agents/.

1.2 What makes analytics agentic?

A plain chatbot answers text with text. An agentic analytics system can:

- translate a business question into an analytical plan,
- call a SQL tool to fetch relevant data,
- call a forecasting, anomaly detection, or KPI tool,
- retrieve definitions and policy rules from company documents,
- inspect intermediate results,
- ask for clarification when the question is ambiguous,
- produce an answer with assumptions, evidence, and recommended next actions.

The important point is that the model is not expected to "know" the company's facts from memory. It should use tools to access current data and documentation.

2 Why Agentic Business Analytics Is a Game Changer

2.1 The old bottleneck

Many companies have invested heavily in data infrastructure. They have dashboards, data warehouses, semantic layers, and reporting pipelines. Yet decision makers still often complain that:

- they cannot find the right dashboard,
- the dashboard answers the first question but not the follow-up question,
- metrics are not consistently defined,
- documentation exists but is not read,
- analysts spend too much time producing repetitive summaries,
- managers want interpretation, not only charts.

The bottleneck is not only data availability. It is the conversion of data into context-aware decisions.

2.2 The new layer: conversational orchestration

Agentic analytics adds a conversational orchestration layer above existing systems. A manager can ask:

> Why did enterprise revenue decline in the Baltic region last week, and is it related to the pricing policy change?

A well-designed agent can break this into sub-questions:

1. Which products and customers contributed most to the decline?
2. Was the decline unusual relative to recent trends?

3. Did the pricing policy change apply to those products or customers?
4. Were there operational issues such as stockouts, delayed invoices, or campaign pauses?
5. What action should the business take?

This is not just a prettier search box. It is a new way to combine data retrieval, business rules, and explanation.

2.3 Where companies gain value

Area	Traditional approach	Agentic approach
KPI monitoring	Static dashboard and manual review	Agent flags anomalies, retrieves context, and writes a first explanation
Executive reporting	Analysts prepare recurring slide text	Agent drafts commentary from verified data and documentation
Self-service BI	Users filter dashboards	Users ask questions in natural language and receive SQL-backed answers
Data literacy	Training sessions and metric glossaries	Agent explains metric definitions and links them to live examples
Root-cause analysis	Analyst manually checks multiple systems	Agent runs a guided investigation across tools
Governance	Policies stored separately from analysis	Agent retrieves relevant rules before recommending action

2.4 Why this matters strategically

Agentic analytics can change the economics of insight. Many business questions are too small to justify a formal analytics project but too important to ignore. Examples include:

- "Which customers should sales contact today?"
- "Why is conversion lower after the checkout redesign?"
- "Which warehouse is causing the shipping delay?"
- "Does this discount comply with our pricing policy?"
- "What should I say in the weekly performance meeting?"

When these questions can be answered quickly and responsibly, analytics becomes more embedded in daily work. The organization moves from periodic reporting to continuous decision support.

3 The Role of Analytics and Analysts

Agentic systems do not remove the need for analysts. They increase the importance of analytical thinking.

3.1 Analysts move up the value chain

In a traditional workflow, analysts often spend time on repetitive tasks:

- rewriting similar SQL queries,
- explaining the same KPI definitions,
- copying numbers into weekly reports,
- answering basic dashboard questions,
- searching internal documentation.

Agents can automate parts of this work. The analyst then spends more time on higher-value responsibilities:

- defining trustworthy metrics,
- designing analytical tools,
- validating outputs,
- building causal explanations,
- evaluating whether recommendations are practical,
- communicating uncertainty and trade-offs.

3.2 The analyst as agent designer

The analyst's role expands from report creator to **decision-system designer**. This includes:

1. deciding which questions the agent should answer,
2. choosing which data sources are safe to expose,
3. writing tool descriptions that make correct use likely,
4. defining guardrails for privacy, compliance, and accuracy,
5. creating test cases from real business scenarios,
6. monitoring failures and improving the system.

3.3 Analytics is still the foundation

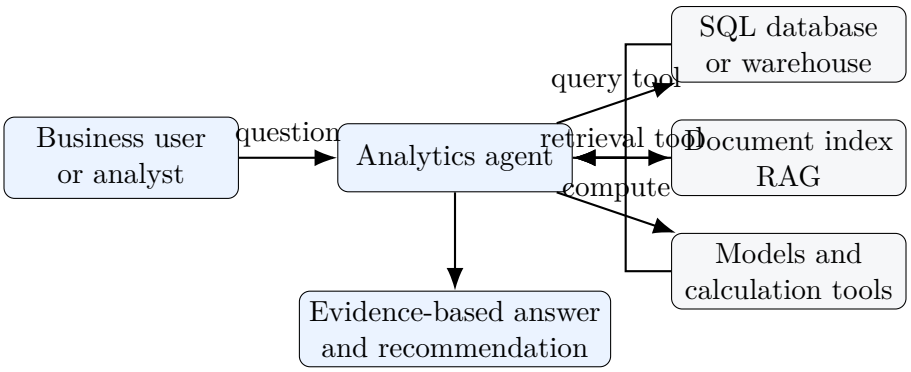
An agent can write a fluent answer even when the analysis is weak. This is why conventional analytics skills remain essential:

- SQL matters because the agent needs reliable data access.
- Statistics matters because uncertainty and variation must be interpreted correctly.
- Data visualization matters because users still need to inspect patterns.
- Experimentation matters because not every correlation is causal.
- Domain knowledge matters because numbers need business meaning.

The future analyst is not replaced by the agent. The future analyst designs, supervises, and improves the agentic analytics system.

4 Architecture of an Agentic Analytics System

4.1 A typical architecture



4.2 Core components

Agent instructions. Define the role, decision style, and boundaries. Example: "You are a business analytics assistant. Use tools for company facts. Do not invent data."

SQL tool. Retrieves current structured data from trusted tables or views.

RAG tool. Retrieves relevant passages from company documentation, such as KPI definitions, pricing policy, product notes, or compliance rules.

Analytical tools. Compute metrics, detect anomalies, run forecasts, estimate churn risk, or calculate scenarios.

Guardrails. Restrict unsafe queries, protect confidential data, validate outputs, and require human approval for high-impact actions.

Tracing and evaluation. Record what the agent did, which tools it used, and whether its final answer was correct.

5 Worked Example: SQL plus RAG Agent

5.1 Scenario

Suppose a company wants an internal assistant for revenue analysis. The assistant should answer questions such as:

> Why did revenue fall in the North region last week? Use the sales data and check whether the pricing policy or product documentation explains the pattern.

The agent needs two kinds of evidence:

1. **Structured data** from a SQL database, for example revenue by region, product, and week.
2. **Unstructured knowledge** from company documents, for example metric definitions, pricing policy, product launch notes, and campaign calendars.

5.2 Database design for the example

A minimal analytics table might look like this:

```
CREATE TABLE sales_weekly (
    week_start DATE,
    region TEXT,
    product TEXT,
    customer_segment TEXT,
    revenue REAL,
    units INTEGER,
    discount_rate REAL
);
```

In a real company, the SQL tool should usually query curated analytics views rather than raw transactional tables. This reduces risk and keeps business definitions consistent.

5.3 Documentation index for RAG

The documentation side can include:

- metric glossary,
- pricing policy,
- product catalog,
- marketing calendar,
- sales playbook,
- compliance rules.

RAG means **retrieval-augmented generation**. Instead of asking the model to answer from memory, we retrieve relevant document chunks and include them as evidence. The agent can then combine live data with retrieved business context.

5.4 Implementation strategy

For a first version, build the system in five steps:

1. Create a read-only SQL query tool.
2. Create a document retrieval tool.
3. Give the agent clear instructions about when to use each tool.
4. Ask the agent to cite which evidence it used.
5. Evaluate the agent on realistic questions.

5.5 A complete teaching example

The code below is intentionally compact, but it shows the main design pattern. In production, you would separate configuration, logging, authorization, and retrieval indexing into separate modules.

```python
"""
analytics_agent.py

Example: an OpenAI Agents SDK business analytics agent that can:
1. Query a SQL database through a controlled tool.
2. Retrieve company documentation through a simple RAG tool.
3. Combine both sources into a business answer.
"""

import asyncio
import json
import sqlite3
from pathlib import Path
from typing import Any

from agents import Agent, Runner, function_tool

DB_PATH = Path("company_analytics.db")
DOCS_PATH = Path("company_docs_chunks.json")


def is_safe_select(query: str) -> bool:
    """Allow only simple read-only SELECT queries for the demo."""
    normalized = " ".join(query.lower().split())
    forbidden = [
        "insert ", "update ", "delete ", "drop ", "alter ",
        "create ", "replace ", "attach ", "detach ", "pragma "
    ]
    return normalized.startswith("select ") and not any(
        word in normalized for word in forbidden
    )


@function_tool
def query_sales_database(sql: str) -> dict[str, Any]:
    """Run a read-only SQL SELECT query against curated sales analytics data.

    Use this tool for revenue, units, discount, region, product, and
    customer-segment questions. The query must be a read-only SELECT statement.
    Prefer aggregation queries and avoid returning unnecessary rows.
    """
    if not is_safe_select(sql):
        return {
            "error": "Only read-only SELECT queries are allowed.",
            "allowed_example": (
                "SELECT region, SUM(revenue) AS revenue "
                "FROM sales_weekly GROUP BY region"
            ),
        }

    with sqlite3.connect(DB_PATH) as conn:
        conn.row_factory = sqlite3.Row
        rows = conn.execute(sql).fetchall()

    return {
        "row_count": len(rows),
        "rows": [dict(row) for row in rows[:100]],
```

```
        "truncated": len(rows) > 100,
    }


@function_tool
def retrieve_company_docs(question: str, top_k: int = 4) -> dict[str, Any]:
    """Retrieve relevant company documentation passages for a business question.

    Use this tool for KPI definitions, pricing rules, product notes,
    campaign explanations, compliance policies, and internal procedures.
    """
    docs = json.loads(DOCS_PATH.read_text())
    query_terms = set(question.lower().split())

    scored = []
    for doc in docs:
        text = (doc["title"] + " " + doc["text"]).lower()
        score = sum(1 for term in query_terms if term in text)
        if score > 0:
            scored.append((score, doc))

    scored.sort(key=lambda item: item[0], reverse=True)
    matches = [doc for _, doc in scored[:top_k]]

    return {
        "matches": matches,
        "note": (
            "This demo uses keyword retrieval. In production, use embeddings "
            "and a vector database or managed file-search service."
        ),
    }


analytics_agent = Agent(
    name="Agentic Business Analytics Assistant",
    instructions="""
You are a careful business analytics assistant.

Rules:
1. Use query_sales_database for numerical claims about revenue, units,
   discounts, regions, products, or customer segments.
2. Use retrieve_company_docs when the question depends on definitions,
   policy, product context, campaign context, or internal rules.
3. Do not invent data. If the tools do not provide enough evidence, say so.
4. Separate facts, interpretation, and recommended actions.
5. Mention important assumptions and data limitations.
6. For SQL, prefer compact aggregation queries.
""",
    tools=[query_sales_database, retrieve_company_docs],
)


async def main():
    question = (
        "Why did revenue fall in the North region last week? "
        "Check the data and relevant pricing or product documentation."
    )
    result = await Runner.run(analytics_agent, question)
```

```
    print(result.final_output)


if __name__ == "__main__":
    asyncio.run(main())
```

5.6 Example documentation chunks

For a classroom demonstration, the file `company_docs_chunks.json` could contain simple chunks:

```
[
  {
    "doc_id": "pricing-2026-01",
    "title": "Pricing Policy 2026",
    "text": "Discounts above 15 percent require approval for enterprise customers. The
     North region started a temporary discount review on May 1."
  },
  {
    "doc_id": "product-alpha-note",
    "title": "Product Alpha Supply Note",
    "text": "Product Alpha had limited availability in the North region during the
    first two weeks of May due to warehouse replenishment delays."
  },
  {
    "doc_id": "metric-glossary-revenue",
    "title": "Metric Glossary: Revenue",
    "text": "Revenue is recognized when invoices are issued, net of discounts but
    before refunds."
  }
]
```

5.7 What the agent should do

For the question about North region revenue, the agent should:

1. query recent North revenue by week and product,
2. compare last week with previous weeks,
3. retrieve relevant pricing and product documents,
4. identify whether the decline is concentrated in a product, segment, or discount pattern,
5. explain the likely drivers,
6. recommend next actions.

For example, the agent might run a query like:

```
SELECT
    week_start,
    product,
    SUM(revenue) AS revenue,
    SUM(units) AS units,
    AVG(discount_rate) AS avg_discount
FROM sales_weekly
WHERE region = 'North'
GROUP BY week_start, product
```

```
ORDER BY week_start DESC, product
LIMIT 40;
```

Then it might retrieve documents using:

```
retrieve_company_docs(
    "North region revenue decline pricing policy product availability"
)
```

5.8 Expected answer structure

A good answer should be structured like this:

1. **Short conclusion.** "Revenue fell mainly because Product Alpha revenue dropped in the North region."
2. **Evidence from SQL.** "Product Alpha accounts for 78 percent of the week-over-week decline."
3. **Evidence from documents.** "The product note says Product Alpha had limited North availability during the first two weeks of May."
4. **Business interpretation.** "The pattern is more consistent with supply availability than weak demand."
5. **Recommended action.** "Check warehouse replenishment status, prioritize backorders, and avoid interpreting the decline as a pricing failure until availability is confirmed."
6. **Limitations.** "The analysis does not include refunds, web traffic, or competitor pricing."

5.9 Why the SQL tool needs guardrails

Giving an agent database access is powerful but risky. The tool should be designed with strong boundaries:

- Use a read-only database user.
- Expose only curated tables or views.
- Block data definition and data modification statements.
- Limit returned rows.
- Log every query.
- Add table-level and column-level permissions.
- Mask personal or sensitive data.
- Prefer metric views that already implement official business definitions.

The demo code blocks obvious non-`SELECT` statements, but production systems need stronger SQL parsing, access control, query timeouts, and audit logs.

5.10 Why the RAG tool matters

Data rarely explains itself. If revenue falls, the explanation may live in a document:

- a pricing policy changed,
- a product was temporarily unavailable,
- a sales campaign ended,
- a metric definition changed,
- a customer segment was reclassified,
- a compliance rule restricts an action.

RAG gives the agent business memory. It lets the agent ground interpretation in internal knowledge rather than guessing.

6 Design Patterns for Agentic Analytics

6.1 Pattern 1: Analyst copilot

The agent helps an analyst work faster. It can draft SQL, explain errors, summarize results, and propose follow-up analysis. The analyst remains in control.

6.2 Pattern 2: Manager self-service assistant

The agent answers common business questions from approved data sources. This works best when the question space is well-defined: sales performance, inventory status, customer churn, campaign monitoring, or KPI definitions.

6.3 Pattern 3: Alert investigator

The system detects an anomaly, then an agent investigates likely causes. For example:

1. anomaly detector flags a revenue drop,
2. agent queries revenue by product and region,
3. agent checks campaign and product notes,
4. agent drafts an alert summary with recommended action.

6.4 Pattern 4: Multi-agent analytics team

For complex settings, several specialist agents can collaborate:

Specialist	Responsibility
SQL analyst agent	Retrieves and summarizes structured data
Forecasting agent	Explains trend, seasonality, and forecast deviations
Marketing agent	Interprets campaign, channel, and conversion effects
Finance agent	Checks margin, budget, and profitability implications
Governance agent	Checks policies, privacy, and compliance rules
Manager agent	Combines specialist outputs into a decision recommendation

Multi-agent systems are useful when expertise areas are genuinely different. They add complexity, so they should not be the first design choice for a simple project.

7 Evaluation: How Do We Know the Agent Is Good?

Agentic analytics systems must be evaluated like analytical products, not only like chatbots.

7.1 Test cases

Create a test set of realistic questions:

- factual KPI questions,
- root-cause questions,
- policy interpretation questions,
- ambiguous questions,
- questions the agent should refuse,
- questions requiring both SQL and RAG.

7.2 Evaluation criteria

Criterion	What to check
Correctness	Are numerical claims supported by database results?
Grounding	Are policy and definition claims supported by retrieved documents?
Tool use	Did the agent call the right tools at the right time?
Completeness	Did the answer cover the main business drivers?
Uncertainty	Did the agent state limitations and assumptions?
Actionability	Are recommendations specific enough to guide work?
Safety	Did the agent avoid exposing sensitive data or taking unauthorized actions?

7.3 Common failure modes

- The agent writes a confident explanation without querying data.
- The SQL query returns the wrong aggregation level.
- The document retrieval tool finds a similar but irrelevant policy.
- The answer confuses correlation with causation.
- The agent over-recommends action from weak evidence.
- The agent exposes row-level sensitive information.

8 Best Practices

8.1 Start small

Begin with one agent and two or three tools. A good first project is a KPI explanation assistant with:

- one curated SQL view,
- one metric glossary retrieval tool,
- one structured answer format.

8.2 Separate retrieval, computation, and interpretation

Do not ask the model to do everything in a prompt. Use tools for facts and calculations:

- SQL retrieves data,
- Python computes metrics,
- RAG retrieves documentation,
- the agent interprets and communicates.

8.3 Make metric definitions explicit

Many analytics errors are definition errors. For example, "revenue" may mean gross revenue, net revenue, booked revenue, recognized revenue, or collected cash. The agent should retrieve the official definition before answering sensitive KPI questions.

8.4 Use human approval for high-impact decisions

Agents are excellent for decision support, but high-impact decisions should include human review. Examples:

- changing prices,
- contacting customers,
- approving credit,
- changing inventory orders,
- sending executive or client-facing reports.

8.5 Log and inspect

Analytics leaders should be able to inspect:

- the user question,
- tool calls,
- SQL queries,
- retrieved documents,
- final answer,
- feedback and corrections.

This is essential for trust, governance, and continuous improvement.

9 Teaching Exercise

9.1 Student task

Build a simple agentic analytics prototype for a fictional retailer. The agent should answer:

> Which product category caused the largest revenue decline this month, and what company documentation might explain it?

9.2 Required components

1. A small SQL database or CSV converted into SQLite.
2. A documentation file with at least five chunks.
3. A SQL tool for aggregated queries.
4. A RAG tool for retrieving relevant documentation.
5. An agent with clear instructions.
6. Three test questions and expected answers.

9.3 Reflection questions

1. When did the agent need data, and when did it need documentation?
2. Which tool call was most important for the final answer?
3. What could go wrong if the SQL tool had no restrictions?
4. How would you evaluate whether the recommendation is reliable?
5. What should remain under human control?

10 Conclusion

Agentic business analytics is a game changer because it changes analytics from a set of static artifacts into an interactive decision-support capability. Dashboards and notebooks are still useful, but agents make it easier to move from question to evidence to recommendation.

The best systems do not treat the language model as an oracle. They treat it as an orchestrator: a reasoning interface that calls tools, retrieves facts, checks documentation, and communicates clearly. SQL provides current structured evidence. RAG provides organizational context. Analysts provide design, validation, judgment, and governance.

For many companies, this combination can reduce repetitive reporting, improve data access, increase data literacy, and make analytics part of everyday decision making. The role of analytics therefore becomes more central, not less: analytics becomes the operating system for evidence-based management.

Further Reading

- OpenAI Agents SDK: https://openai.github.io/openai-agents-python/
- Agents SDK tools guide: https://openai.github.io/openai-agents-python/tools/
- Running agents: https://openai.github.io/openai-agents-python/running_agents/
- Sessions: https://openai.github.io/openai-agents-python/sessions/