Building Semantic Views with Cortex Analyst - Banking Edition

Overview

This guide walks you through building semantic views with Cortex Analyst, specifically adapted for banking data. Based on the College of AI Data Agent HOL methodology, we'll create business-friendly semantic models that enable natural language querying of your banking data.

What You'll Learn:

- Create semantic models for banking data
- Define business-friendly vocabulary and relationships
- Set up verified queries for consistent results
- Enable natural language access to banking insights

Phase 1: Banking Semantic Model Setup

Step 1: Understanding Semantic Models

Why Semantic Models Matter for Banking

Without semantic models, Cortex Analyst struggles with:

- **Complex banking terminology** (NPL rates, AUM, credit utilization)
- **Multi-table relationships** (customers → accounts → transactions → loans)
- **Business context** (fiscal periods, risk tiers, customer segments)
- **Consistent metrics** (everyone using the same definitions)

With semantic models, you get:

- **Business-friendly vocabulary** that maps "client" to "customer_name"
- **Explicit relationships** between banking entities
- **Time and fact dimensions** for proper date handling
- **Verified queries** that ensure consistent results

Banking Data Structure

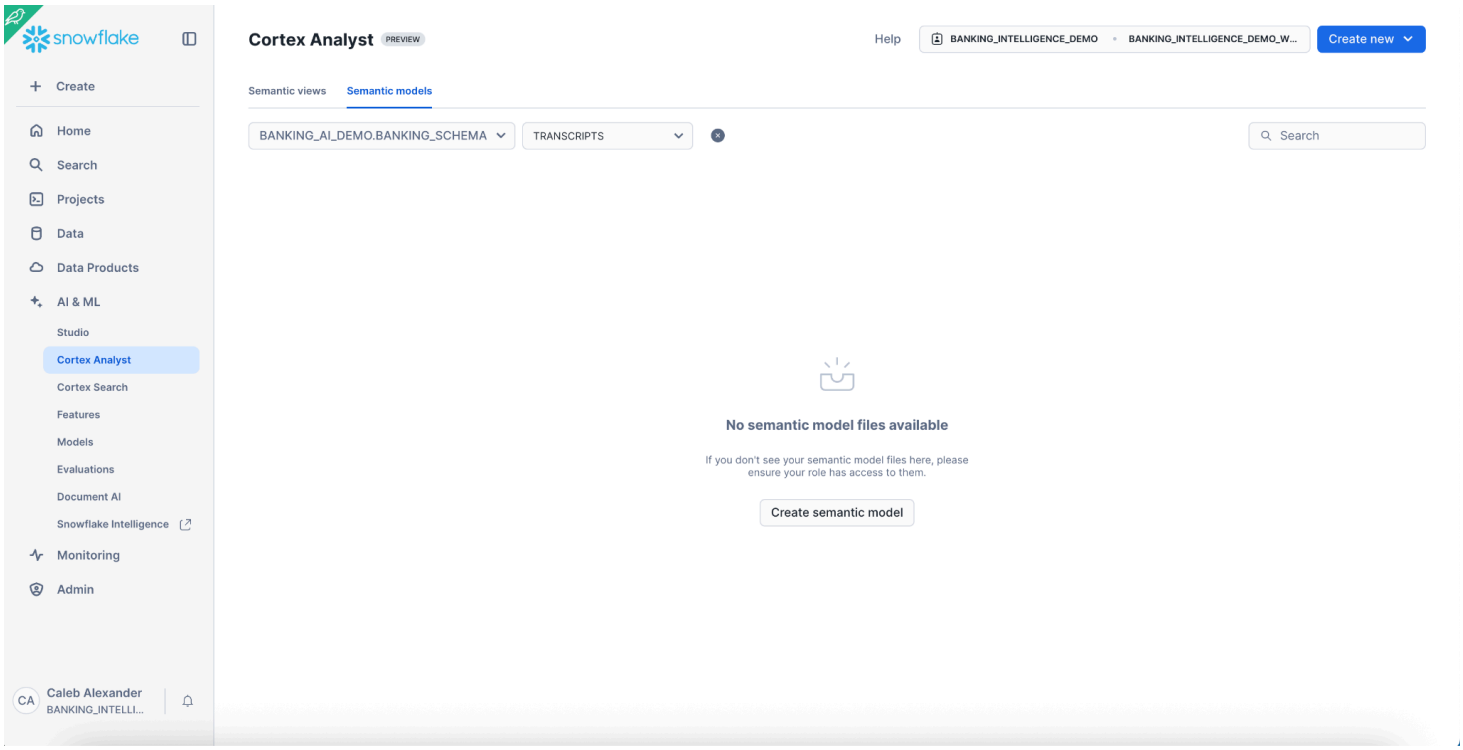
Our banking demo includes these core tables:

- customer_dim - Customer information and segmentation
- account_dim - Account details and balances
- transaction_fact - Transaction history and activity
- loan_portfolio_fact - Loan data with risk metrics

Step 2: Create Banking Semantic Model

Navigate to Cortex Analyst

1. In Snowflake Snowsight, switch to your BANKING_INTELLIGENCE_DEMO role.
2. In the left sidebar, expand **AI & ML** and click **Studio**.
3. Select **Cortex Analyst**.
4. Select **Create new model** option.

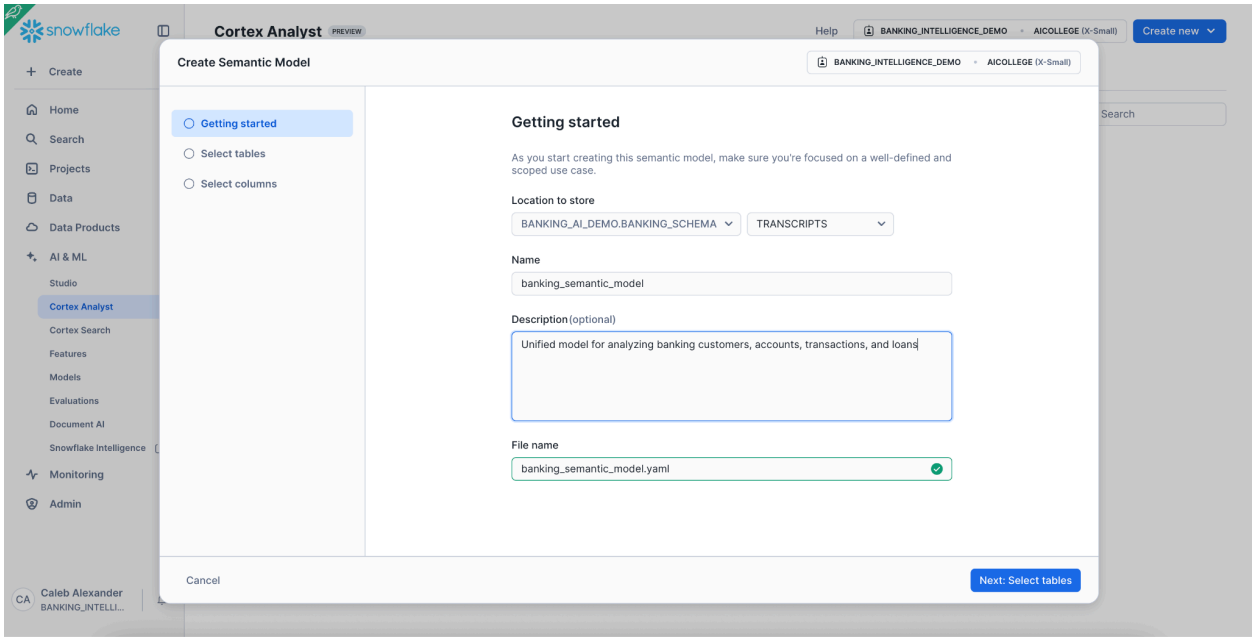


Choose Semantic Model Type

For development and banking demos, choose **Stages**:

- **Location**: BANKING_AI_DEMO.BANKING_SCHEMA.TRANSCRIPTS
- **Name**: banking_semantic_model
- **Description**: Unified model for analyzing banking customers, accounts, transactions, and loans

- **File name**: banking_semantic_model.yaml

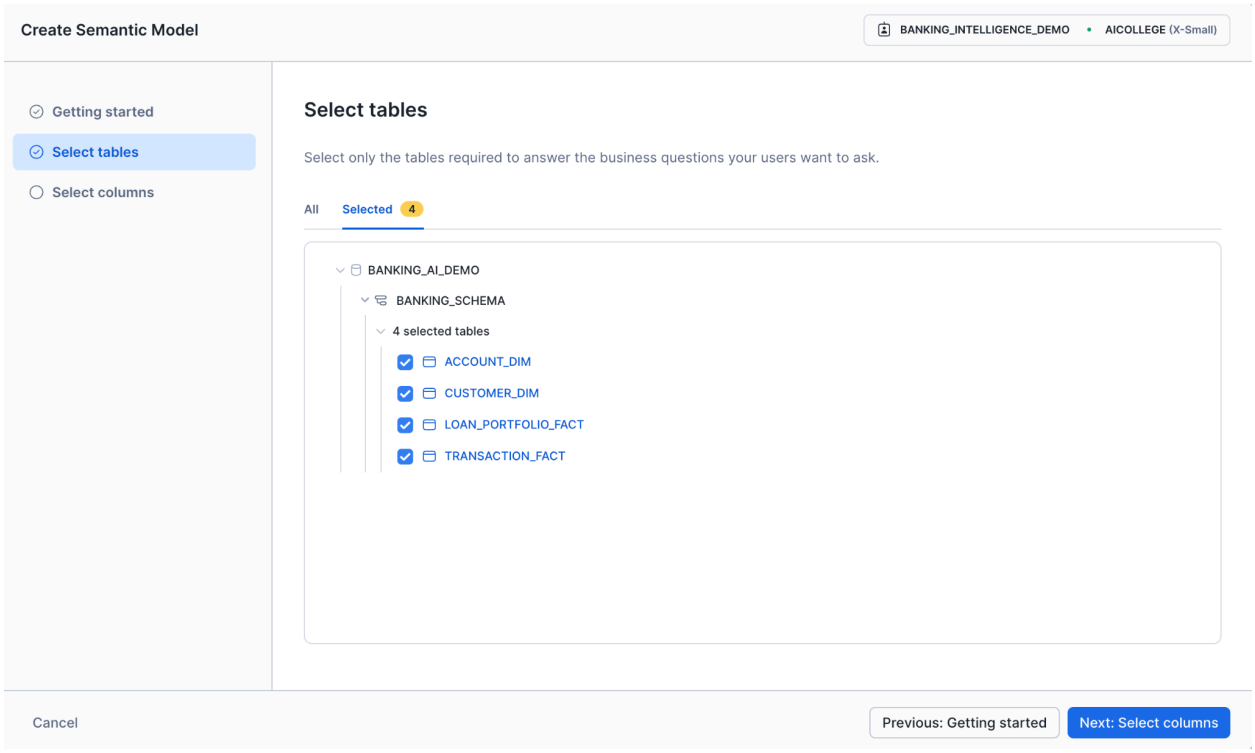


Select Banking Tables

Expand BANKING_AI_DEMO > BANKING_SCHEMA and select:

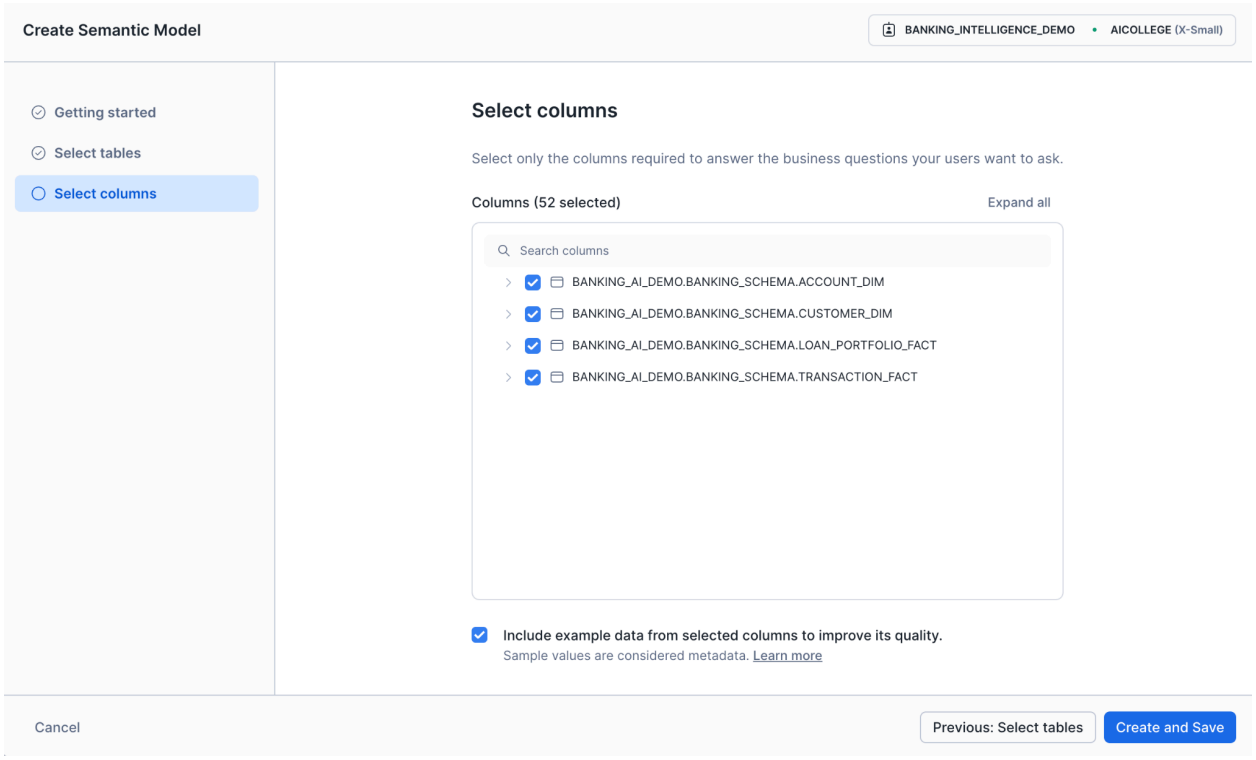
- customer_dim
- account_dim
- transaction_fact

- loan_portfolio_fact



Select All Columns

Include all columns from each table to provide comprehensive access to banking data.



Step 3: Define Banking Business Vocabulary

Add Table Synonyms

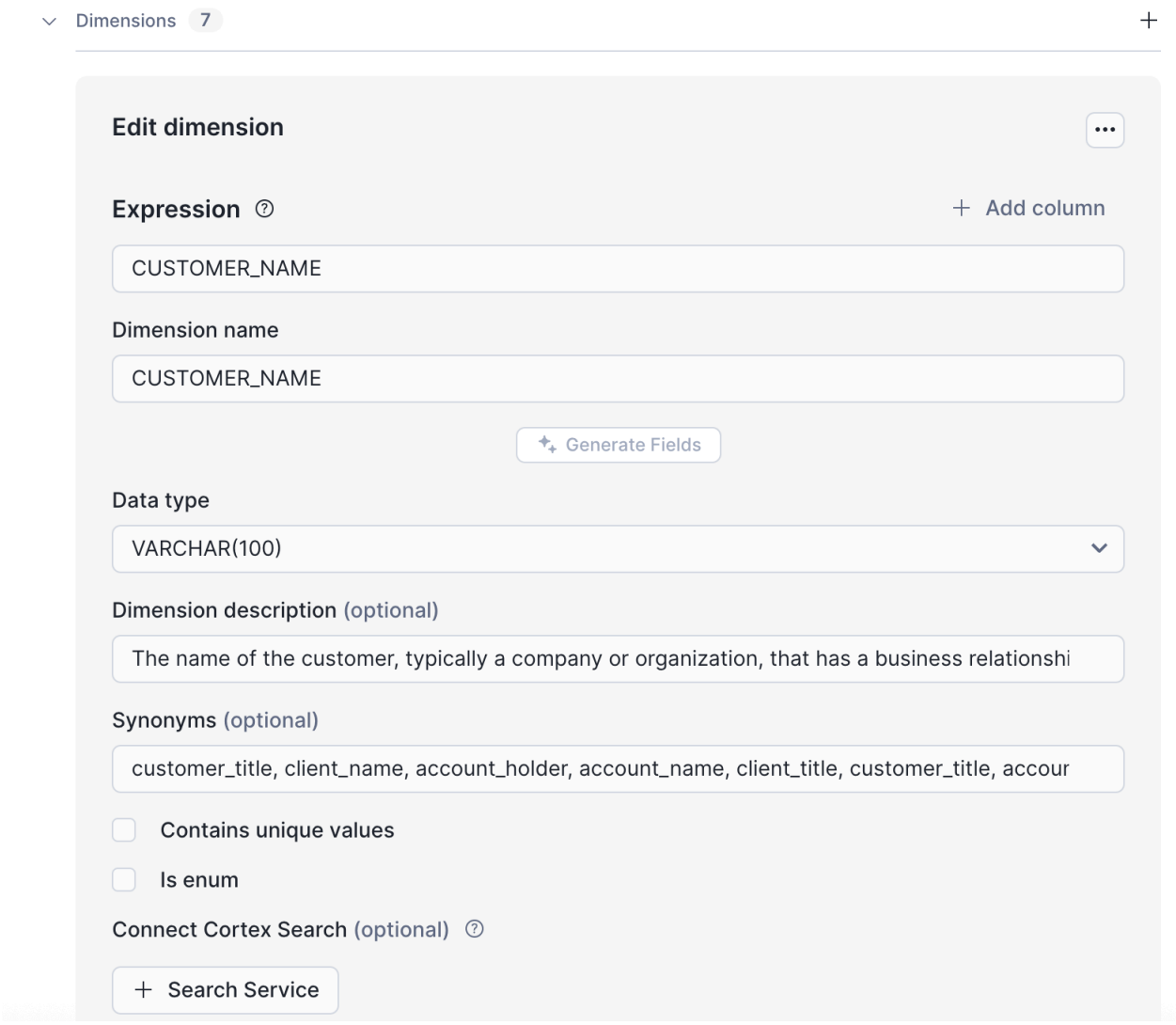
Add business-friendly synonyms for each banking table:

Logical Table	Description	Suggested Synonyms
customer_dim	Customer information and segmentation	customers, clients, account_holders, customer_data
account_dim	Account details and balances	accounts, banking_accounts, customer_accounts, account_data
transaction_fact	Transaction history and activity	transactions, transaction_history, banking_transactions, activity_log
loan_portfolio_fact	Loan data with risk metrics	loans, loan_portfolio, lending_data, credit_data

Add Column Synonyms

In the "Edit dimension" window for each column, you'll find a "Synonyms" field.

- **You might see some synonyms already pre-filled** by Cortex Analyst's automatic detection. This is normal!
- **Your task is to *add to* or *verify* these lists**, ensuring all the business synonyms suggested below are included. The more synonyms, the better Cortex Analyst can understand your natural language questions.



Customer Dimensions:

- customer_name → client_name, account_holder, customer
- customer_type → client_type, customer_segment, account_type
- customer_segment → segment, tier, customer_category
- risk_tier → risk_level, credit_risk, risk_category

Account Dimensions:

- account_type → product_type, account_product, banking_product
- balance → account_balance, current_balance, funds
- product_line → business_line, product_category, service_line

Transaction Dimensions:

- transaction_type → transaction_category, activity_type, transaction_kind
- amount → transaction_amount, transfer_amount, payment_amount
- channel → transaction_channel, payment_method, access_channel

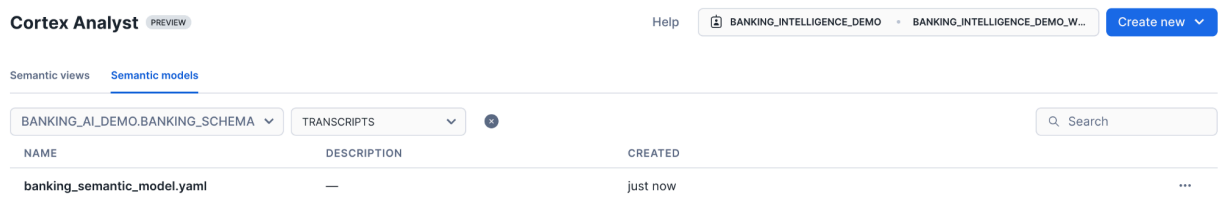
Loan Dimensions:

- loan_amount → credit_amount, borrowed_amount, principal
- outstanding_balance → remaining_balance, unpaid_balance, loan_balance
- interest_rate → rate, loan_rate, credit_rate

- Click **SAVE** in the top right corner of the page

🎯 Phase 2: Testing Your Banking Semantic Model

Step 1: Verify Model Creation

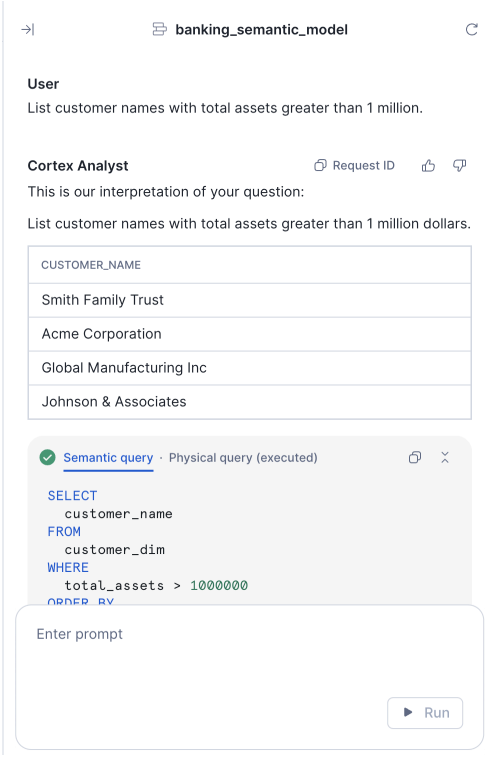


Step 2: Test Basic Banking Queries

Try these natural language questions in Cortex Analyst:

Customer Questions:

- How many customers do we have by segment?
- List customer names with total assets greater than 1 million.
- Count our high value customers.
- What is the average credit score by customer type?

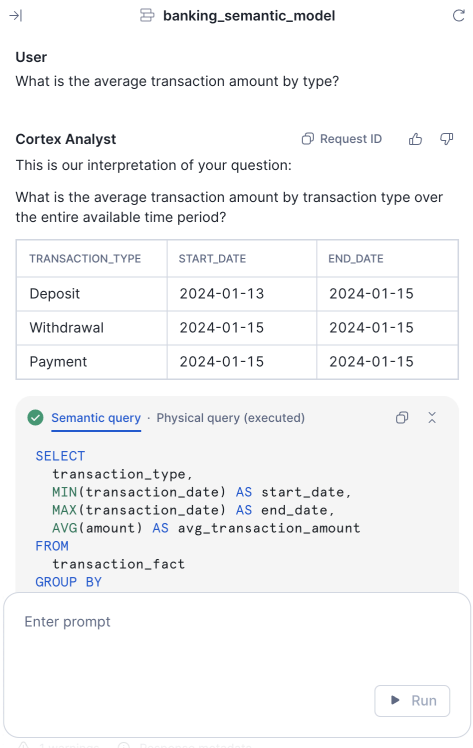


Account Questions:

- What is the total balance across all accounts?
- Show me account distribution by type.
- Which account managers have the most accounts?

Transaction Questions:

- What is the total transaction volume by channel?
- Show me the largest transactions this month.
- What is the average transaction amount by type?



Loan Questions:

- What is our total loan exposure by risk tier?
- What is the customer utilization rate for each customer?
- Which customers have a loan utilization rate above 80 percent?
- What is the average loan size by region?

Best Practices for Banking Semantic Models

1. Use Clear Business Terminology

- Map technical column names to business-friendly terms
- Include industry-specific banking vocabulary
- Add synonyms for common variations

2. Define Proper Relationships

- Use appropriate join types (INNER vs LEFT)
- Specify cardinality (one-to-many, many-to-one)
- Ensure referential integrity

3. Create Meaningful Metrics

- Focus on banking KPIs and performance indicators
- Include both simple counts and complex calculations
- Add business context to metric descriptions

4. Implement Business Rules

- Use custom instructions for consistent formatting
- Define standard filters for common scenarios
- Enforce data governance policies

5. Test Thoroughly

- Verify queries work as expected
- Test edge cases and error conditions
- Validate business logic accuracy

Common Banking Use Cases

Customer 360 Analysis

- Complete customer view across all products
- Relationship depth and profitability
- Risk assessment and credit exposure

Portfolio Management

- Asset allocation and diversification
- Performance tracking and benchmarking
- Risk-adjusted returns

Regulatory Reporting

- Capital adequacy and liquidity ratios
- Risk concentration analysis
- Compliance monitoring and alerts

Operational Analytics

- Transaction volume and patterns
- Channel utilization and preferences
- Service quality and efficiency metrics

Troubleshooting Banking Semantic Models

Common Issues and Solutions

Issue: "Cortex Analyst doesn't understand banking terms"

Solution: Add more synonyms and business vocabulary to your model

Issue: "Queries return incorrect results"

Solution: Verify relationships and join conditions are properly defined

Issue: "Performance is slow on large datasets"

Solution: Add appropriate filters and limit result sets

Issue: "Date calculations are incorrect"

Solution: Ensure proper date formatting and fiscal period definitions

---

Next Steps

1. Extend Your Model

- Add more banking-specific metrics and filters
- Include additional data sources (market data, external feeds)
- Create specialized views for different business lines

2. Integrate with BI Tools

- Connect semantic models to Tableau, Power BI, or Looker
- Create dashboards based on semantic model definitions
- Enable self-service analytics for business users

3. Implement Governance

- Set up role-based access controls
- Define data quality rules and validation
- Establish change management processes

4. Scale Your Solution

- Extend to production data sources
- Add real-time data integration
- Implement automated model updates

Congratulations! You've successfully built a banking semantic model with Cortex Analyst. Users can now ask natural language questions about banking data and get accurate, consistent answers without writing SQL!