

CoCo-Financial Fraud Analytics - Cortex Code Hands-on Lab

Overview

Welcome to the **CoCo-Financial Fraud Analytics** Hands-on Lab! In this lab, you will learn how to use **Cortex Code CLI** to build a complete fraud analytics solution for a fictional financial services company.

CoCo-Financial is a mid-sized financial institution that processes millions of transactions daily. They need to detect and prevent fraudulent activities across their customer base. You'll help them by:

1. Setting up secure authentication to Snowflake
 2. Deploying a comprehensive fraud analytics dataset
 3. Creating semantic views for natural language queries
 4. Building custom Cortex Code skills for fraud analysis
-

Learning Objectives

- By the end of this lab, you will be able to:
- Install and configure Cortex Code CLI
 - Set up Key-Pair or PAT authentication with Snowflake
 - Deploy databases, tables, views, and semantic views using Cortex Code
 - Use built-in Cortex Code skills
 - Create custom skills for domain-specific workflows
 - Query data using natural language through semantic views
-

Time Estimates

Section	With Pre-work Done	Without Pre-work
Prerequisites	0 min	15-20 min
Module 1: Environment Setup	5 min	10 min
Module 2: Authentication	10 min	15 min
Module 3: Deploy Dataset	15 min	15 min
Module 4: Using Skills	10 min	10 min
Module 5: Build Custom Skill	15 min	15 min
Module 6: Experimentation	15 min	15 min
Total	~60 min	~80-90 min

Prerequisites

Before starting this lab, ensure you have the following:

Required

1. **Snowflake Account** with permissions to create objects (Database, Schema, Tables)
2. **Snowflake CLI** (snow) installed
3. **Cortex Code CLI** (cortex) installed
4. **macOS (Apple Silicon or Intel)** or **Linux** operating system
 - Note: Windows is not currently supported; use WSL if on Windows

Quick Prerequisite Check

Run the prerequisite check script to verify your environment:

```
cd /path/to/CoCo_H0L
bash scripts/prereq_check.sh
```

Don't Have a Snowflake Account?

Sign up for a **free Cortex Code trial** at:

https://signup.snowflake.com/cortex-code?utm_cta=pushdown-signup

This provides you with a Snowflake account pre-configured for Cortex Code.

Accelerated Setup (If Prerequisites Not Met)

If you don't have the prerequisites installed, run the accelerated setup script:

```
bash scripts/accelerated_setup.sh
```

This script will:

- Install Snowflake CLI (if not present)
- Install Cortex Code CLI (if not present)
- Guide you through initial configuration

Module 1: Environment Setup

Step 1.1: Verify Prerequisites

Open a terminal and run:

```
# Check Snowflake CLI
snow --version
```

```
# Check Cortex Code CLI
cortex --version
```

Expected output should show version numbers for both tools.

Step 1.2: Test Snowflake Connection

If you already have a Snowflake connection configured:

```
# List existing connections
snow connection list

# Test your connection
snow connection test -c <your_connection_name>
```

Step 1.3: Test Object Creation Permissions

Before proceeding, verify you can create objects in your Snowflake account.

Launch Cortex Code and run a test:

```
cortex
```

In Cortex Code, enter:

```
Test if I can create a database called COCO_FINANCIAL_TEST,
then drop it if successful. Show me the results.
```

If successful: You're ready to proceed to Module 2.

If you get a permission error: You have two options:

1. **Contact your Snowflake administrator** to grant CREATE DATABASE privileges
2. **Create a free trial account** at: https://signup.snowflake.com/cortex-code?utm_cta=pushdown-signup

Module 2: Authentication Setup

This module covers two authentication methods. Choose **one** based on your preference:

- **Option A: Key-Pair Authentication** - More secure, recommended for production
- **Option B: PAT (Programmatic Access Token)** - Simpler setup, good for development

Option A: Key-Pair Authentication

Key-pair authentication uses RSA public/private keys for secure, passwordless authentication. We'll use **Cortex Code** to generate everything for us!

Step 2A.1: Launch Cortex Code

```
cortex
```

Step 2A.2: Generate RSA Key Pair with Cortex Code

In Cortex Code, enter this prompt:

```
Generate an RSA key pair for Snowflake authentication:

1. Create the directory ~/.snowflake/keys if it doesn't exist
2. Generate a 2048-bit RSA private key in PKCS#8 format (unencrypted for
this lab)
3. Save it to ~/.snowflake/keys/rsa_key.p8
4. Generate the corresponding public key
5. Save it to ~/.snowflake/keys/rsa_key.pub
6. Set secure permissions (600) on the private key
7. Display the public key content (without headers) that I'll need for
Snowflake
```

Cortex Code will execute the necessary commands and show you the public key content.

Step 2A.3: Assign Public Key to Your Snowflake User

Still in Cortex Code, enter:

```
Take the public key you just generated and create the SQL command
to assign it to my Snowflake user. My username is: YOUR_USERNAME

Then help me run this command in Snowflake.
```

Cortex Code will generate and execute the ALTER USER command for you.

Step 2A.4: Configure Connection with Cortex Code

In Cortex Code, enter:

```
Create a Snowflake connection configuration for key-pair authentication:

- Connection name: coco_lab
- Account: YOUR_ACCOUNT_IDENTIFIER
- Username: YOUR_USERNAME
```

```
- Private key path: ~/.snowflake/keys/rsa_key.p8
- Warehouse: COMPUTE_WH
- Database: COCO_FINANCIAL
- Schema: FRAUD_ANALYTICS
- Role: YOUR_ROLE
```

Add this to my ~/.snowflake/connections.toml file.

Step 2A.5: Test the Connection

In Cortex Code:

```
Test my new coco_lab connection to make sure key-pair authentication is
working.
```

Or exit and test manually:

```
snow connection test -c coco_lab
```

Congratulations! You just used Cortex Code to set up secure key-pair authentication without memorizing any OpenSSL commands!

Option B: PAT (Programmatic Access Token) Authentication

PAT tokens are easier to set up and are ideal for development and testing.

Step 2B.1: Generate PAT via Snowsight

1. Log into Snowsight
2. Click your username (bottom-left)
3. Select **My Profile**
4. Go to **Authentication** tab
5. Click **+ Programmatic access token**
6. Configure:
 - **Name:** coco_lab_token
 - **Expiration:** 30 days (or as needed)
 - **Role restriction:** (optional) select your role
7. Click **Generate**
8. **IMPORTANT:** Copy and save the token immediately - it won't be shown again!

Alternative: Generate PAT with Cortex Code

Launch Cortex Code and enter:

```
Generate a Programmatic Access Token (PAT) for my Snowflake user.  
My username is: YOUR_USERNAME  
Token name: coco_lab_token  
Expiration: 30 days
```

```
Run the ALTER USER command to create the token.
```

IMPORTANT: Copy and save the token immediately - it won't be shown again!

Step 2B.2: Configure Connection with Cortex Code

In Cortex Code, enter:

```
Create a Snowflake connection using PAT authentication:
```

- Connection name: coco_lab_pat
- Account: YOUR_ACCOUNT_IDENTIFIER
- Username: YOUR_USERNAME
- PAT token: PASTE_YOUR_TOKEN_HERE
- Warehouse: COMPUTE_WH
- Database: COCO_FINANCIAL
- Schema: FRAUD_ANALYTICS
- Role: YOUR_ROLE

```
Add this to my ~/.snowflake/connections.toml file.  
Also mention the security best practice for storing tokens.
```

Step 2B.3: Test the Connection

In Cortex Code:

```
Test my coco_lab_pat connection to verify PAT authentication is working.
```

Or manually:

```
snow connection test -c coco_lab_pat
```

Module 3: Deploy Fraud Analytics Dataset

Now let's deploy the CoCo-Financial fraud analytics dataset using Cortex Code.

Step 3.1: Launch Cortex Code with Your Connection

```
# Use your configured connection
cortex -c coco_lab
# OR for PAT
cortex -c coco_lab_pat
```

Step 3.2: Review the Dataset Schema

Before deploying, review the schema documentation:

```
Read the file data/schema.md and explain the CoCo-Financial
fraud analytics data model to me.
```

Step 3.3: Deploy the Database and Tables

In Cortex Code, enter:

```
Deploy the CoCo-Financial fraud analytics dataset using the SQL
in scripts/deploy_fraud_dataset.sql. Create all objects and
load the sample data. Show me the progress.
```

Alternatively, deploy step by step:

```
Create a database called COCO_FINANCIAL with a schema called
FRAUD_ANALYTICS.
Then create the following tables based on the schema in data/schema.json:
- CUSTOMERS
- ACCOUNTS
- TRANSACTIONS
- MERCHANTS
- FRAUD_LABELS
- ALERTS
```

```
After creating the tables, generate and load 10,000 sample records
with realistic fraud patterns.
```

Step 3.4: Create Views

```
Create the following views in COCO_FINANCIAL.FRAUD_ANALYTICS:
```

1. VW_TRANSACTION_SUMMARY – Aggregates transactions by customer with fraud indicators
2. VW_FRAUD_METRICS – Daily fraud metrics including fraud rate, total flagged amount, and alert counts

Step 3.5: Create Semantic View

```
Create a semantic view called SV_FRAUD_ANALYTICS based on the YAML
definition in data/semantic_view.yaml. This will enable natural
language queries against our fraud data.
```

Step 3.6: Verify Deployment

```
Show me a summary of all objects in the COCO_FINANCIAL database
including row counts for each table.
```

Module 4: Working with Cortex Code Skills

Skills are reusable workflows that teach Cortex Code how to complete specific tasks consistently.

Step 4.1: List Available Skills

In Cortex Code:

```
List all available skills and briefly describe what each does.
```

Or use the shortcut:

```
$$
```

Step 4.2: Understand Skill Locations

Skills are loaded from multiple locations in priority order:

Priority	Location	Path
1	Project	<code>.cortex/skills/</code> in current directory
2	Global	<code>~/.snowflake/cortex/skills/</code>
3	Remote	Configured in <code>skills.json</code>
4	Bundled	Shipped with Cortex Code

Step 4.3: Explore a Bundled Skill


```
Show me the structure of the synthetic-data-demo skill.  
What are its main workflow steps?
```

Step 4.4: Use a Skill

Let's use the data-governance skill to analyze our fraud dataset:

```
$data-governance analyze the access patterns and permissions  
for the COCO_FINANCIAL database.
```

Step 4.5: Understand Skill Structure

A skill consists of:

```
skill-name/  
├── SKILL.md           # Main skill file (required)  
├── templates/        # Optional templates  
└── references/       # Optional reference docs
```

The **SKILL.md** file contains:

- **Frontmatter:** name, description, triggers
- **Workflow:** Step-by-step instructions
- **Stopping Points:** Where to pause for user input
- **Output:** Expected deliverables

Module 5: Build a Custom Skill

Now let's create a custom skill for CoCo-Financial fraud analysis.

Step 5.1: Navigate to the Skill Directory

The skill template is already created at:

```
CoCo_H0L/skill/coco-financial-fraud/SKILL.md
```

Step 5.2: Review the Skill

```
Read the skill file at skill/coco-financial-fraud/SKILL.md  
and explain what it does.
```

Step 5.3: Test the Custom Skill

First, let's add the skill to your project's skill path:

```
# Create local skills directory
mkdir -p .cortex/skills

# Copy our custom skill
cp -r skill/coco-financial-fraud .cortex/skills/
```

Step 5.4: Use the Custom Skill

In Cortex Code:

```
$coco-financial-fraud analyze recent transaction patterns
and identify potential fraud indicators in our dataset.
```

Step 5.5: Modify the Skill (Optional)

```
Help me enhance the coco-financial-fraud skill to also include:
1. Velocity checks (rapid successive transactions)
2. Geographic anomaly detection
3. Time-based pattern analysis
```

Module 6: Experimenting with Cortex Code

This module lets you explore various Cortex Code capabilities hands-on.

6.1: Natural Language SQL Generation

Try these queries against your fraud dataset:

```
What are the top 10 merchants by transaction volume?
```

```
Show me customers with more than 5 flagged transactions
in the last 30 days.
```

```
What's the average transaction amount by merchant category?
```

6.2: Data Exploration

Describe the TRANSACTIONS table and show me sample data with any interesting patterns.

Find correlations between transaction amount and fraud likelihood.

6.3: Semantic View Queries

Use natural language with your semantic view:

Using the SV_FRAUD_ANALYTICS semantic view, answer:
What was the fraud rate last month compared to the previous month?

Using the semantic view, show me the highest risk customer segments.

6.4: Code Generation

Write a Python script that connects to Snowflake and generates a fraud risk report for the top 100 highest-risk customers.

Create a Streamlit dashboard that displays real-time fraud metrics from our COCO_FINANCIAL database.

6.5: Documentation Generation

Generate technical documentation for the COCO_FINANCIAL database including an ERD diagram and data dictionary.

6.6: SQL Optimization

Analyze this query and suggest optimizations:

```
SELECT c.customer_name, COUNT(t.transaction_id) as txn_count,  
       SUM(CASE WHEN f.is_fraud = TRUE THEN 1 ELSE 0 END) as fraud_count  
FROM CUSTOMERS c
```

```
JOIN ACCOUNTS a ON c.customer_id = a.customer_id
JOIN TRANSACTIONS t ON a.account_id = t.account_id
LEFT JOIN FRAUD_LABELS f ON t.transaction_id = f.transaction_id
GROUP BY c.customer_name
ORDER BY fraud_count DESC;
```

6.7: Working with Files

Reference files directly in your prompts:

```
Read @data/schema.md and create a presentation summary
of our data model.
```

6.8: Keyboard Shortcuts

Try these shortcuts in Cortex Code:

Shortcut	Action
@	File completion - reference local files
\$	Skill tagging - invoke skills
#	Snowflake table reference
!	Run bash command
/	Slash commands menu
?	Quick help
Shift+Tab	Cycle operational modes
Ctrl+J	Insert newline

6.9: Operational Modes

Cortex Code has three operational modes:

- 1. **Accept Edits** (default) - Normal mode with permission checks
- 2. **Plan Mode** (/plan) - Review actions before execution
- 3. **Bypass Mode** (/bypass) - Auto-approve all tools (use carefully)

Try switching modes:

```
/plan
```

Then run a query to see how plan mode shows you actions before executing.

6.10: Session Management

```
/status          # Show current session status
/model           # Switch AI model
/clear           # Clear conversation context
```

Cleanup

When you're done with the lab, clean up the resources:

```
-- Run in Snowflake or via Cortex Code
DROP DATABASE IF EXISTS COCO_FINANCIAL;
DROP DATABASE IF EXISTS COCO_FINANCIAL_TEST;
```

To remove your PAT token:

```
ALTER USER YOUR_USERNAME REMOVE PROGRAMMATIC ACCESS TOKEN NAME =
'coco_lab_token';
```

Troubleshooting

Connection Issues

Error	Cause	Fix
Connection failed	Invalid credentials	Verify account, user, and auth method
JWT token invalid	Key mismatch	Re-assign public key to user
PAT expired	Token expired	Generate new PAT token
Permission denied	Insufficient role	Contact admin or use trial account

Common Fixes

```
# List and test connections
snow connection list
snow connection test -c <connection_name>

# Check Cortex Code version
cortex --version

# Reinstall Cortex Code if needed
curl -Ls https://ai.snowflake.com/static/cc-scripts/install.sh | sh
```

Getting Help

- In Cortex Code: Type `?` or `/help`
 - Documentation: <https://docs.snowflake.com/en/user-guide/cortex-code/cortex-code-cli>
 - Trial signup: https://signup.snowflake.com/cortex-code?utm_cta=pushdown-signup
-

Summary

Congratulations! You've completed the CoCo-Financial Fraud Analytics Hands-on Lab. You learned how to:

- Set up Key-Pair and PAT authentication with Snowflake
- Deploy a complete fraud analytics dataset using Cortex Code
- Create and use semantic views for natural language queries
- Work with built-in and custom Cortex Code skills
- Explore various Cortex Code capabilities

Next Steps

1. **Explore more skills:** Check `~/ .snowflake/cortex/skills/` for additional skills
 2. **Build your own skills:** Use the skill-development skill to create custom workflows
 3. **Connect to your data:** Apply these techniques to your own datasets
 4. **Share with your team:** Skills can be shared via Git repositories
-

Additional Resources

- [Cortex Code CLI Documentation](#)
 - [Key-Pair Authentication Guide](#)
 - [Programmatic Access Tokens](#)
 - [Semantic Views Documentation](#)
 - [Snowflake CLI](#)
-

Lab Version: 1.0

Last Updated: February 2025

Author: CoCo-Financial HOL Team