PROJECT DOCUMENTATION

Investment Opportunity Analysis – USGS Cobalt Mining Database

Student: Gladys Jacob
Course: CSE 385
Date: December 07, 2025
Professor: Professor John Mattox

---

1. OVERVIEW

Purpose

This project creates a normalized relational database from USGS cobalt mining data to support investment opportunity analysis. The database includes geological occurrences, resource estimates, production records, and reference data for mining sites across the United States.

Data Source

United States Geological Survey (USGS) Cobalt Mining Database

- Original format: 11 CSV files
- Contains information on 37 mining sites across multiple states

Citation:

Burger, M.H., Schmeda, G., Long, K.R., Reyes, T.A., and Karl. N.A., 2018, Cobalt Deposits in the United States: U.S. Geological Survey data release, https://doi.org/10.5066/P9V74HIU.

---

2. EXTERNAL LIBRARIES & CITATIONS

Required Software

- MySQL Server (I used MySQL Workbench)
- Java JDK 17+

- Eclipse IDE (or any Java IDE)
- MySQL Connector/J (JDBC driver)

Libraries Used

- MySQL JDBC Driver
  - Oracle Corporation. (2024). *MySQL Connector/J 8.0 Developer Guide*.

Java Standard Libraries (Built-In)

- `javax.swing` – GUI
- `java.awt` – Layout & graphics
- `java.sql` – JDBC
- `java.io` – File handling

No external graphics libraries were used (GUI uses only Swing components).

---

3. INSTALLATION INSTRUCTIONS

Step 1 - Import the Complete Database

The file CobaltMining_CompleteDB.sql automatically:

- Creates all 25 tables
- Inserts all cleaned & normalized data

Instructions:

1. Open MySQL Workbench → connect to local MySQL
2. File → Open SQL Script → select `CreateTables.sql` and execute
3. File → Open SQL Script → select `CobaltMining_CompleteDB.sql` and execute

Verification:

```
USE cobalt_mining;

SHOW TABLES;              -- Should show 25 tables
```

```
SELECT COUNT(*) FROM SITE;     -- Expect 37

SELECT COUNT(*) FROM REFERENCE; -- Expect 117

SELECT COUNT(*) FROM LOCATION_POINT; -- Expect 71

SELECT COUNT(*) FROM RESOURCE; -- Expect 290

SELECT COUNT(*) FROM PRODUCTION; -- Expect 20
```

Step 2: Install Stored Procedures

1. Open and execute: **StoredProcedures.sql**

Verification:

```
SHOW PROCEDURE STATUS WHERE Db = 'cobalt_mining';
```

You should see:

- filter_geol_by_commodity_type
- clean_resource_indicators

Test Database:

```
CALL filter_geol_by_commodity_type('cobalt', 'Deposit');

CALL clean_resource_indicators();
```

Both should return results without errors.

Step 3: Run Java GUI Application

1. Create new Java project in Eclipse: `CobaltMiningApp`
2. Uncheck "Use default location", select the folder from extracted ZIP
3. Add MySQL Connector/J JAR to build path:
   - Right-click project → Build Path → Configure Build Path
   - Libraries → Add External JARs → Select mysql-connector-j JAR
4. Update `DatabaseConnection.java` line 12:

   ```
   private static final String DB_PASSWORD = "your_mysql_password";
   ```

5. Run `CobaltMiningGUI.java`:

    - Right-click file → Run As → Java Application

---

4. USAGE INSTRUCTIONS

The GUI opens with four main tabs:

1. DASHBOARD TAB

- Statistics Cards (4 cards displayed)
    - Each card has a “More information” button → pop-up dialogue with stats
- Key Insights Panel
    - Most tracked commodity
    - Average resources per site
    - Data completeness
- Refresh Button
    - Reloads all statistics
    - Confirmation popup appears
    - Use after modifying/refreshing data

2. TABLE VIEWER TAB

Select Table:

- Location Points, Geological Occurrences, Location Polygons, Production Records, Resource Estimates

Load Table:

- Reconstructs original USGS structure through joins
- Populates scrollable, sortable table
- Bottom bar shows row count

Help Button (i):
Explains:

- Sorting rules
- Color coding in Resource table
- Why negative values appear in Mat_Amnt column

Table Features:

- Sortable columns (click any header)
    - Click any column header to sort by that column (ascending or descending)
    - Small arrow appears in header showing sort direction
- Scrollable (use mouse wheel or scrollbar)
- Color-coded (Resources table only)
    - Dark Green: High-values investment potential (>100,000)
    - Light Green: High values (>10,000)
    - Light Yellow: Medium values (>1,000)
    - White: Low values or missing data (<1,000)

3. GEOLOGICAL FILTER TAB

Filters geological occurrences by:

- Commodity (cobalt, copper, nickel, gold, silver, zinc, lead)
- Feature Type (Deposit, Prospect, Occurrence, Mine, Mining District)

Apply Filter:

- Calls stored procedure
- Shows results with status message
- Table supports sorting & scrolling

Clear Results:

- Empties table for new search

4. RESOURCE CLEANER TAB

Executes stored procedure to clean all resource records.

Load Cleaned Resources:

- Shows cleaned dataset (contained amounts, units, etc.)
- Fully sortable & scrollable

Clear Button:

- Empties table

NOTE:

All tabs (except dashboard) include a refresh and search button

- Refresh reloads data directly from database
- Search works across ALL columns and filters automatically as you type
    - Clear search field to show all results again

---

5. ADDITIONAL NOTES

Data Import Process:

Before producing the final SQL export, I wrote:

- `CreateTables.sql` — defined schema
- `DatabaseImporter.java` — parsed and cleaned raw CSVs

Note:
You do *not* need to re-run the importer.
`CobaltMining_CompleteDB.sql` already includes the finished, normalized database

.