MINING BLOCK MODEL ANALYSIS: CATEGORISING ORE AND WASTE BLOCKS USING PYTHON AND SQL

TABLE OF CONTENTS

1. INTRODUCTION..... 1
1.1 Business Context And Problem..... 1
1.2 Project Objectives..... 1
2. DATA PREPARATION (Python)..... 2
2.1. Dataset description..... 2
2.2. Tools and technologies..... 2
2.3. Creating Business_Category And Exporting CSV..... 2
3. MAPPING..... 2
4. SQL ANALYSIS (POSTGRESQL)..... 2
4.1. Loading The Cleaned Data..... 2
4.2. QA Checks (Counts And Mapping)..... 3
4.3. Profitability By Category And Rock Type..... 3
4.4. Key Block-Level Insights (Top/Bottom And Grade Buckets)..... 4
5. CONCLUSION..... 6
5.1. Main Findings..... 6

1. INTRODUCTION

1.1 Business Context And Problem

The mining company manages a large open-pit block model with tens of thousands of blocks, each described by coordinates, rock type, grade, tonnage, costs, and profit. In its raw form, the data is technical and difficult for business stakeholders to use when comparing ore and waste performance. The key problem is to group these blocks into simple, business-friendly categories (ore vs waste) and understand how profit and other economic variables behave across those categories to support mine planning decisions.

1.2 Project Objectives

This project aims to transform the raw block model into a cleaned and summarised dataset that is easier to analyse. The specific objectives are to:

- Detect and handle outliers in key economic fields such as profit, before loading the data into the database.
- Create a new business_category field from Rock_Type to distinguish ore blocks from waste material.
- Use SQL in PostgreSQL to summarise block counts and profitability by category and rock type, and to identify the most profitable and least profitable blocks

2. DATA PREPARATION (Python)

2.1. Dataset description

The dataset is an open-pit mining block model containing 75,000 rows and 13 columns, including block ID, spatial coordinates (X, Y, Z), Rock_Type, ore grade percentage, tonnage, ore value per tonne, mining and processing costs, a waste flag, profit, and a binary target field. All columns are complete, with no missing values, allowing the analysis to focus on distribution and outlier behaviour rather than imputation.

2.2. Tools and technologies

Data cleaning and feature creation were performed in Python using a Google Colab notebook with the pandas library for tabular data manipulation. PostgreSQL and pgAdmin were used to store the cleaned dataset and run SQL queries, providing a realistic database environment similar to what is used in production analytics workflows

2.3. Creating Business_Category And Exporting CSV

In Python, outliers in profit and other numeric fields were identified using the interquartile range (IQR), and extreme profit values were removed while preserving realistic variation in grade and ore value. A business_category column was then created by mapping ore-bearing rock types (Hematite and Magnetite) to Ore_Block and Waste to Waste_Material, turning technical geological labels into business-friendly groups. The cleaned DataFrame was exported as mining_cleaned.csv, ready for import into PostgreSQL for SQL analysis.

3. MAPPING

To simplify the information for business stakeholders, we defined a new business_category field based on the original Rock_Type values. Instead of working directly with multiple rock types, the data model groups blocks into two high-level categories that demonstrate how they are treated in mine planning and financial analysis.

Rock_Type	business_category
Hematite	Ore_Block
Magnetite	Ore_Block
Waste	Waste_Material

The business_category field maps ore-bearing rock types (Hematite and Magnetite) to Ore_Block and assigns Waste to Waste_Material, so that technical geological codes are converted into clearer business labels for downstream SQL analysis and reporting.

4. SQL ANALYSIS (POSTGRESQL)

4.1. Loading The Cleaned Data

Aim: Create a structured table in PostgreSQL and load the cleaned CSV so analysis can run in SQL instead of in flat files.

```
DROP TABLE IF EXISTS mining_block_model;
CREATE TABLE mining_block_model (
  block_id VARCHAR(10) PRIMARY KEY ,
  x integer,
  y integer,
  z integer,
  rock_type text,
  ore_grade_pct numeric,
  tonnage numeric,
  ore_value_y_per_t numeric,
  mining_cost_y numeric,
  processing_cost_y numeric,
  waste_flag integer,
  profit_y numeric,
  target integer,
  business_category text
);

SELECT * FROM mining_block_model;
SELECT COUNT(*) FROM mining_block_model;
```

4.2. QA Checks (Counts And Mapping)

Aim: Verify that the business_category mapping is correct and that block counts by category and rock type look sensible.

```
-- Check mapping worked
SELECT rock_type,
       business_category,
       COUNT(*) AS block_count
FROM mining_block_model
```

```
GROUP BY rock_type, business_category
ORDER BY rock_type;

-- Size of each business category
SELECT business_category,
       COUNT(*) AS total_blocks
FROM mining_block_model
GROUP BY business_category;

-- Blocks count by rock type inside each business category
SELECT business_category,
       rock_type,
       COUNT(*) AS block_count
FROM mining_block_model
GROUP BY business_category, rock_type
ORDER BY business_category, block_count DESC;
```

4.3. Profitability By Category And Rock Type

Aim: Summarise how profit behaves for ore vs waste and for each rock type to answer the business question “which groups make or lose money?”.

```
-- Profit behaviour by category
SELECT business_category,
       AVG(profit_y) AS avg_profit,
       SUM(profit_y) AS total_profit
FROM mining_block_model
GROUP BY business_category
ORDER BY avg_profit DESC;

-- Profit by rock type
SELECT rock_type,
       business_category,
       AVG(profit_y) AS avg_profit,
       SUM(profit_y) AS total_profit,
       COUNT(*)      AS block_count
FROM mining_block_model
GROUP BY rock_type, business_category
ORDER BY total_profit DESC;

-- Waste impact summary
SELECT
    COUNT(*)      AS waste_blocks,
    SUM(profit_y) AS total_waste_profit,
    AVG(profit_y) AS avg_waste_profit
FROM mining_block_model
WHERE business_category = 'Waste_Material';
```

4.4. Key Block-Level Insights (Top/Bottom And Grade Buckets)

Aim: Drill down from aggregates to individual blocks and grade groups to highlight high-value opportunities and high-risk losses.

```sql
-- High-grade ore blocks
SELECT block_id,
       rock_type,
       business_category,
       ore_grade_pct,
       profit_y
FROM mining_block_model
WHERE business_category = 'Ore_Block'
  AND ore_grade_pct > 40
ORDER BY ore_grade_pct DESC
LIMIT 20;

-- Most profitable blocks
SELECT block_id, rock_type, business_category, profit_y
FROM mining_block_model
ORDER BY profit_y DESC
LIMIT 10;

-- Most unprofitable blocks
SELECT block_id, rock_type, business_category, profit_y
FROM mining_block_model
ORDER BY profit_y ASC
LIMIT 10;

-- Top 5 profitable Hematite blocks
SELECT block_id,
       rock_type,
       business_category,
       profit_y
FROM (
    SELECT block_id,
           rock_type,
           business_category,
           profit_y,
           ROW_NUMBER() OVER (
               PARTITION BY rock_type
               ORDER BY profit_y DESC
           ) AS rn
    FROM mining_block_model
) t
WHERE rock_type = 'Hematite'
  AND rn <= 5
ORDER BY profit_y DESC;

-- Top 5 profitable Magnetite blocks
SELECT block_id,
       rock_type,
       business_category,
       profit_y
FROM (
    SELECT block_id,
```

```
            rock_type,
            business_category,
            profit_y,
            ROW_NUMBER() OVER (
                PARTITION BY rock_type
                ORDER BY profit_y DESC
            ) AS rn
    FROM mining_block_model
) t
WHERE rock_type = 'Magnetite'
  AND rn <= 5
ORDER BY profit_y DESC;

-- Profit vs ore grade for ore blocks
SELECT
    CASE
        WHEN ore_grade_pct < 20 THEN 'Low_grade'
        WHEN ore_grade_pct < 40 THEN 'Medium_grade'
        ELSE 'High_grade'
    END AS grade_bucket,
    COUNT(*)      AS block_count,
    AVG(profit_y) AS avg_profit,
    SUM(profit_y) AS total_profit
FROM mining_block_model
WHERE business_category = 'Ore_Block'
GROUP BY grade_bucket
ORDER BY avg_profit DESC;
```

5. CONCLUSION

5.1. Main Findings

The analysis shows that the new business_category field successfully simplifies the block model into two clear business groups: Ore_Block and Waste_Material. In PostgreSQL, ore blocks display positive average and total profit, while waste blocks show negative profitability, proving that the mapping aligns with real mining economics. Aggregations by rock type and grade buckets highlight that high-grade Hematite and Magnetite blocks contribute most of the value, and the window-function queries identify specific blocks that represent the best opportunities..