India Companies Database Documentation

By Adnan Rasool

Overview

This documentation describes a SQLite database system designed to store and analyze financial data for Indian companies across multiple years (2023-2024). The system provides functionality for data storage, comparison analysis, and generation of industry insights through SQL views. The database is structured to handle complex financial metrics while maintaining data integrity and enabling efficient analysis of company performance trends.

Database Schema

Tables

1. companies

Primary table storing company information. This table serves as the central reference point for all company-related data.

- `company_id` (INTEGER): Primary key, auto-incrementing identifier for each company
- `name` (TEXT): Company name (UNIQUE) - Enforced unique constraint prevents duplicate company entries
- `industry` (TEXT): Industry classification for sector-based analysis
- `headquarters` (TEXT): Company headquarters location for geographical distribution analysis

2. financial_data

Stores yearly financial metrics for each company. This table contains the core financial performance data with temporal dimension.

- `financial_id` (INTEGER): Primary key, auto-incrementing identifier for each financial record
- `company_id` (INTEGER): Foreign key referencing companies table, ensures data integrity
- `year` (INTEGER): Financial year of the data
- `rank` (INTEGER): Company rank within the dataset
- `revenue` (REAL): Annual revenue in standardized currency
- `profit` (REAL): Annual profit/loss in standardized currency
- `assets` (REAL): Total assets valuation
- `market_value` (REAL): Current market valuation
- `revenue_growth` (REAL): Year-over-year revenue growth percentage
- `state_controlled` (BOOLEAN): Indicates government ownership status
- `forbes_rank` (INTEGER): Global Forbes ranking if available

3. industry_categories

Reference table for standardizing industry classifications across the database.

- `industry_id` (INTEGER): Primary key, auto-incrementing identifier for each industry
- `industry_name` (TEXT): Standardized industry name (UNIQUE)

  This table enables consistent industry categorization and facilitates sector-based analysis.

4. cities

Reference table for city classifications and tiering.

- `city_id` (INTEGER): Primary key, auto-incrementing identifier for each city
- `city_name` (TEXT): City name (UNIQUE)

- `tier` (TEXT): City tier classification (e.g., Tier 1, Tier 2, Tier 3)

  This table supports geographical analysis and city-tier based comparisons.

Views

1. yoy_comparison

This view provides comprehensive year-over-year comparison of company performance, enabling trend analysis and growth tracking.

```
CREATE VIEW yoy_comparison AS
SELECT
    c.name,
    c.industry,
    c.headquarters,
    f1.year as year_2024,
    f1.revenue as revenue_2024,
    f1.profit as profit_2024,
    f2.year as year_2023,
    f2.revenue as revenue_2023,
    f2.profit as profit_2023,
    ((f1.revenue - f2.revenue) / f2.revenue * 100) as revenue_growth,
    ((f1.profit - f2.profit) / ABS(f2.profit) * 100) as profit_growth
FROM companies c
JOIN financial_data f1 ON c.company_id = f1.company_id AND f1.year = 2024
JOIN financial_data f2 ON c.company_id = f2.company_id AND f2.year = 2023
```

Key features:

- Calculates revenue and profit growth percentages
- Handles negative profit values correctly using ABS function
- Joins current and previous year data for direct comparison
- Includes company metadata for contextual analysis

2. industry_summary

This view aggregates financial metrics at the industry level, providing sector-wide insights and benchmarks.

```
CREATE VIEW industry_summary AS
SELECT
    c.industry,
    f.year,
    COUNT(*) as company_count,
    SUM(f.revenue) as total_revenue,
    SUM(f.profit) as total_profit,
    AVG(f.revenue) as avg_revenue,
    AVG(f.profit) as avg_profit
FROM companies c
JOIN financial_data f ON c.company_id = f.company_id
GROUP BY c.industry, f.year
```

Key features:

- Calculates industry-wide totals and averages
- Provides company count per industry
- Groups data by both industry and year for temporal analysis
- Enables industry performance benchmarking

Core Functions

Database Initialization

```
def initialize_database():
    create_tables()
    populate_database()
    create_views()
```

This function orchestrates the complete database setup process:

1. Creates all necessary tables with appropriate constraints and relationships
2. Populates the database with initial data from CSV sources
3. Establishes analytical views for data analysis
4. Performs necessary integrity checks during setup

Data Population Process

```
def populate_database():
    # Process 2024 data
    for _, row in df_2024.iterrows():
        company_id = insert_company(row['Name'], row['Industry'], row['Headquarters'])
        insert_financial_data(company_id, row, 2024)

    # Process 2023 data
    for _, row in df_2023.iterrows():
        company_id = insert_company(row['Name'], row['Industry'], row['Headquarters'])
        insert_financial_data(company_id, row, 2023)
```

The population process handles:

1. Sequential processing of CSV data years
2. Automatic company record creation or updating
3. Financial data insertion with appropriate year tagging
4. Maintenance of referential integrity between tables

Company Insertion

```
def insert_company(name, industry, headquarters):
    cursor.execute('''
    INSERT OR IGNORE INTO companies (name, industry, headquarters)
    VALUES (?, ?, ?)
    ''', (name, industry, headquarters))
```

```
    cursor.execute('SELECT company_id FROM companies WHERE name = ?', (name,))
    return cursor.fetchone()[0]
```

This function:

- Handles new company insertion
- Prevents duplicate company entries
- Returns existing company ID if company already exists
- Maintains data consistency across years

Financial Data Insertion

```
def insert_financial_data(company_id, data, year):
    cursor.execute('''
    INSERT OR REPLACE INTO financial_data
    (company_id, year, rank, revenue, profit, assets, market_value,
     revenue_growth, state_controlled, forbes_rank)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
    ''', (
        company_id,
        year,
        data.get('Rank'),
        data.get('Revenue'),
        data.get('Profit'),
        data.get('Assets'),
        data.get('Value'),
        data.get('Revenue_Growth'),
        data.get('State_Controlled', False),
        data.get('Forbes_Rank')
    ))
```

This function manages:

- Insertion of yearly financial metrics
- Handling of missing data through .get() method
- Default values for optional fields
- Update of existing records when necessary

Example Queries and Analysis

1. Top 10 Companies by Revenue (2024)

```
SELECT c.name, f.revenue, f.profit, c.industry
FROM companies c
JOIN financial_data f ON c.company_id = f.company_id
WHERE f.year = 2024
ORDER BY f.revenue DESC
LIMIT 10
```

This query provides:

- Revenue and profit analysis of top performers
- Industry context for leading companies
- Basis for market leadership analysis

2. Most Profitable Industries

```
SELECT
    industry,
    year,
    total_profit,
    avg_profit
FROM industry_summary
ORDER BY total_profit DESC
```

This analysis enables:

- Industry profitability comparison
- Sector performance tracking
- Identification of high-performing industries

3. Companies with Highest Revenue Growth

```
SELECT
    name,
    industry,
    revenue_growth,
    revenue_2024,
    revenue_2023
FROM yoy_comparison
ORDER BY revenue_growth DESC
LIMIT 10
```

This query identifies:

- Fastest-growing companies
- Year-over-year performance changes
- Industry growth patterns

Data Source Requirements

The system requires two primary CSV files:

- `cleaned_india_companies_2024.csv`
- `cleaned_india_companies_2023.csv`

Required columns structure:

- Name: Company name (text)
- Industry: Industry classification (text)
- Headquarters: Company location (text)
- Rank: Company ranking (integer)
- Revenue: Annual revenue (float)

- Profit: Annual profit (float)
- Assets: Total assets (float)
- Value: Market value (float)
- Revenue_Growth: YoY growth (float)
- State_Controlled: Government ownership (boolean, optional)
- Forbes_Rank: Global ranking (integer, optional)

Data validation and formatting:

- Numerical values should be in consistent units
- Text fields should be properly cleaned and standardized
- Missing values should be handled appropriately
- Dates should be in standard format