ABT Analytics Dashboard

Setup Guide

Technical Documentation

Contents

1 What This Is

This is a high-performance analytics dashboard built for ABT Corporation to analyze transaction data. The backend is written in Go and the frontend uses React with TypeScript. It can handle large CSV files (tested with 5M+ records) and provides real-time insights through charts and tables.

The whole thing loads in under 10 seconds even with massive datasets, thanks to concurrent processing and smart caching.

2 Tech Stack

Here's what I used to build this:

- **Backend**: Go 1.23 with Gorilla Mux for routing
- **Frontend**: React 18 + TypeScript + Vite
- **UI**: Chakra UI for components, Recharts for visualizations
- **Data Processing**: Concurrent Go routines with configurable worker pools
- **Caching**: In-memory + file-based caching system
- **Build**: Make for automation, Docker for deployment

3 Before You Start

Make sure you have these installed:

Required Software:

- Go 1.23 or later
- Node.js 18 or later
- npm (comes with Node.js)
- Make (optional, but recommended)

System Requirements:

- At least 2GB RAM (4GB+ recommended for large datasets)
- 1GB+ free disk space
- Multi-core CPU recommended

4 Quick Setup (Easiest Way)

I've created a script that does everything for you:

```
# Clone the repo
git clone https://github.com/randilt/abt-corp-analytics-dashboard.git
cd abt-corp-analytics-dashboard

# Make the script executable and run it
```

```
chmod +x start.sh
./start.sh
```

This script will:

1. Install all Go dependencies
2. Build the backend
3. Install frontend dependencies
4. Build the React app
5. Start both services

Once it's done, you can access:

- Dashboard: `http://localhost:4173`
- API: `http://localhost:8080/api/v1`
- Health check: `http://localhost:8080/health`

5 Manual Setup

If you want to do it step by step or the script doesn't work:

5.1 Backend Setup

```
# From project root
go mod download
go build -o bin/server cmd/server/main.go
./bin/server

# Or use make (if you have it)
make build
make run
```

5.2 Frontend Setup

```
cd web
npm install
npm run build
npm run preview

# For development mode:
npm run dev
```

6 Dataset Setup

Important: You need to replace the dummy data with your actual dataset.
The app expects your CSV file at `data/raw/transactions.csv`

```
data/
          raw/
                transactions.csv    # Your actual data goes here
          processed/
              analytics_cache.json # Generated automatically
```

Your CSV should have these columns:

`transaction_id, transaction_date, user_id, country, region, product_id, product_name, category, price, quantity, total_price, stock_quantity, added_date`

The included `transactions.csv` only has 100 sample rows for testing.

7 Configuration

You can tweak the app behavior with environment variables:

7.1 Performance Tuning

```
# For large datasets, you might want to reduce these
CSV_BATCH_SIZE=5000 CSV_WORKER_POOL=4 ./bin/server

# For powerful machines, increase them
CSV_BATCH_SIZE=20000 CSV_WORKER_POOL=16 ./bin/server

# Other useful settings
SERVER_PORT=9090 ./bin/server
LOG_LEVEL=debug ./bin/server
```

7.2 All Available Options

Variable	Default
SERVER_PORT	8080
CSV_FILE_PATH	./data/raw/transactions.csv
CSV_BATCH_SIZE	10000
CSV_WORKER_POOL	8
CACHE_TTL	24h
LOG_LEVEL	info

8 API Endpoints

The backend provides these endpoints:

Pagination example:

```
# First 50 records
curl "http://localhost:8080/api/v1/analytics/country-revenue?limit=50&
    offset=0"

# Next 50
curl "http://localhost:8080/api/v1/analytics/country-revenue?limit=50&
    offset=50"
```

Method	Endpoint	What it does
GET	`/api/v1/analytics`	All dashboard data
GET	`/api/v1/analytics/stats`	Summary stats
GET	`/api/v1/analytics/country-revenue`	Country revenue (paginated)
GET	`/api/v1/analytics/top-products`	Top 20 products
GET	`/api/v1/analytics/monthly-sales`	Monthly trends
GET	`/api/v1/analytics/top-regions`	Top 30 regions
POST	`/api/v1/analytics/refresh`	Refresh cache

9 Testing

```
# Run all tests
make test

# Just unit tests
make test-unit

# Get coverage report (opens in browser)
make test-coverage
```

The coverage report will be at `./coverage/coverage.html`

10 Docker (Optional)

If you prefer Docker:

```
# Build and run with Docker Compose
docker-compose up -d

# Or build manually
docker build -t analytics-dashboard .
docker run -p 8080:8080 -v $(pwd)/data:/app/data analytics-dashboard

# Or use the pre-built image
docker pull randilt/abt-corp-analytics-dashboard:v1
```

11 Common Issues

11.1 Backend won't start

- Check if port 8080 is available: `lsof -i :8080`
- Make sure your CSV file exists and is readable
- Try: `go version` to verify Go installation

11.2 Data not loading

- Verify CSV format matches the expected columns
- Check file permissions on the data directory
- Try refreshing cache: `curl -X POST http://localhost:8080/api/v1/analytics/refresh`
- For very large files, reduce `CSV_BATCH_SIZE`

11.3 Frontend build fails

- Clear node modules: `rm -rf node_modules && npm install`
- Check Node.js version: `node --version`
- Make sure you're in the `web` directory

11.4 Performance issues

- Reduce batch size and worker pool for less RAM usage
- Increase them for faster processing on powerful machines
- Check memory usage at: `http://localhost:8080/health`

12 Performance Notes

- Fresh load with 5M records: 8-10 seconds
- Cached responses: Nearly 5ms
- Memory usage: Nearly 500MB for full cache
- Concurrent processing with configurable workers

For huge datasets (10M+ records), you might want to tune the batch size and worker count based on your machine specs.

13 What's Included

The dashboard shows exactly what was requested:

1. **Country-Level Revenue Table** - Sortable, paginated table with country, product, revenue, and transaction count
2. **Top 20 Products** - Bar chart showing most purchased products with stock levels
3. **Monthly Sales Volume** - Line chart highlighting peak sales months
4. **Top 30 Regions** - Bar chart of highest revenue regions with items sold