# 🚀 AiiA to Supabase Migration Guide

## 📋 Migration Overview

This guide will help you migrate your local PostgreSQL database to Supabase using the generated SQL script.

## 🔧 Prerequisites

- Supabase account (free tier available)
- Access to your local PostgreSQL database
- `pg_dump` and `psql` tools installed

## ⚡ Quick Migration Steps

### 1. Create Supabase Project

1. Go to supabase.com (https://supabase.com)
2. Create new project
3. Wait for database setup (2-3 minutes)
4. Note your database connection details

### 2. Get Supabase Connection Details

From your Supabase dashboard:
- **Database URL**: `postgresql://postgres:[password]@db.[project-ref].supabase.co:5432/postgres`
- **Direct URL**: Same as above
- **Project Reference**: Found in Project Settings → General
- **Database Password**: Set during project creation

### 3. Create Database Schema

Run the migration SQL in Supabase:

**Option A: Using Supabase SQL Editor (Recommended)**
1. Open Supabase Dashboard → SQL Editor
2. Copy contents of `supabase-migration.sql`
3. Paste and run the entire script
4. Verify all tables are created

**Option B: Using psql Command Line**

```
psql "postgresql://postgres:[password]@db.[project-ref].supabase.co:5432/postgres" \
  -f supabase-migration.sql
```

### 4. Export Data from Local PostgreSQL

Export your existing data:

```
# Export all data (excluding schema since we already created it)
pg_dump --host=localhost \
        --username=your_username \
        --dbname=your_database_name \
        --data-only \
        --no-owner \
        --no-privileges \
        --file=aiia_data_export.sql

# Alternative: Export specific tables if needed
pg_dump --host=localhost \
        --username=your_username \
        --dbname=your_database_name \
        --table='"User"' \
        --table='"Portfolio"' \
        --table='"Watchlist"' \
        --table='"Trade"' \
        --data-only \
        --no-owner \
        --no-privileges \
        --file=aiia_core_data.sql
```

## 5. Import Data to Supabase

Import the exported data:

```
# Import data to Supabase
psql "postgresql://postgres:[password]@db.[project-ref].supabase.co:5432/postgres" \
  -f aiia_data_export.sql
```

## 6. Update Environment Variables

Update your `.env` file:

```
# Replace with your Supabase connection strings
DATABASE_URL="postgresql://postgres:[password]@db.[project-ref].supabase.co:5432/post-
gres"
DIRECT_URL="postgresql://postgres:[password]@db.[project-ref].supabase.co:5432/post-
gres"

# Update NextAuth URL if needed
NEXTAUTH_URL="https://your-app-domain.com"
```

## 7. Update Prisma Configuration

Update `prisma/schema.prisma`:

```
datasource db {
  provider  = "postgresql"
  url       = env("DATABASE_URL")
  directUrl = env("DIRECT_URL")
}
```

## 8. Verify Migration

Test the migration:

```
# Generate Prisma client with new connection
npx prisma generate

# Test database connection
npx prisma db pull

# Optional: Push any schema changes
npx prisma db push
```

## 🛡️ Security Features Included

### Row Level Security (RLS)

The migration includes RLS policies to ensure:
- Users can only access their own data
- Public access to market data and news
- Secure isolation between user accounts

### Database Indexes

Optimized indexes for:
- User lookups and authentication
- Portfolio and trading queries
- Market data retrieval
- Real-time notifications

### Triggers

Automatic timestamp updates for:
- User profile changes
- Portfolio updates
- System settings

## 📊 Database Schema Summary

### Core Tables

- **User Management**: User, Account, Session, VerificationToken
- **Trading**: Portfolio, Trade, Watchlist, Alert
- **Market Data**: MarketData, AIAnalysis, NewsArticle
- **Communication**: Notification, ChatMessage
- **Business**: Subscription, FinancialDisclaimer, SystemSetting

### Key Features

- **Full NextAuth.js support** for authentication
- **Comprehensive trading system** with order types
- **AI analysis integration** with confidence scoring
- **Real-time market data** with caching
- **Multi-tier subscription** system
- **Compliance tracking** with financial disclaimers

# 🔍 Troubleshooting

## Common Issues

### Connection Errors

```
# Test connection
psql "postgresql://postgres:[password]@db.[project-ref].supabase.co:5432/postgres" -c "
\l"
```

**Import Errors**
- Check for conflicting data types
- Verify foreign key constraints
- Ensure proper escaping of special characters

**RLS Issues**
- Disable RLS temporarily during data import:

```
ALTER TABLE "User" DISABLE ROW LEVEL SECURITY;
-- Import data
ALTER TABLE "User" ENABLE ROW LEVEL SECURITY;
```

## Verification Queries

```
-- Check table creation
SELECT table_name FROM information_schema.tables
WHERE table_schema = 'public'
ORDER BY table_name;

-- Check data import
SELECT
  (SELECT COUNT(*) FROM "User") as users,
  (SELECT COUNT(*) FROM "Portfolio") as portfolios,
  (SELECT COUNT(*) FROM "Trade") as trades;

-- Test RLS policies
SELECT * FROM "User" LIMIT 1;
```

# 🚀 Post-Migration Steps

1. **Update Application**
   - Deploy with new environment variables
   - Test authentication flow
   - Verify trading functionality

2. **Monitor Performance**
   - Check query performance in Supabase dashboard
   - Monitor API usage and limits
   - Set up alerts for errors

3. **Backup Strategy**
   - Enable point-in-time recovery in Supabase

- Set up regular backups
- Test restore procedures

## 📞 Support

- **Supabase Documentation**: supabase.com/docs (https://supabase.com/docs)
- **PostgreSQL Migration Guide**: postgresql.org/docs (https://postgresql.org/docs)
- **Prisma with Supabase**: supabase.com/docs/guides/integrations/prisma (https://supabase.com/docs/guides/integrations/prisma)

## ✅ Migration Checklist

- [ ] Supabase project created
- [ ] Database schema deployed (`supabase-migration.sql`)
- [ ] Local data exported
- [ ] Data imported to Supabase
- [ ] Environment variables updated
- [ ] Prisma configuration updated
- [ ] Application tested with new database
- [ ] RLS policies verified
- [ ] Performance monitoring set up
- [ ] Backup strategy implemented

**Migration Complete!** 🎉