Database Connection Configuration Guide

Connection Modes

1. Direct Connection (Port 5432)

Best for: Development, lower latency **Cons:** No connection pooling, limited concurrent connections

```bash
export DB_HOST="db.wxinjyotnrqxrwqrtvkp.supabase.co"
export DB_PORT="5432"
export DB_NAME="postgres"
export DB_USER="postgres"
export DB_PASSWORD="jovpeW-pukgu0-nifron"
```

2. Connection Pooler - Transaction Mode (Port 6543)

Best for: Production, many short queries **Pros:** Connection pooling, handles many clients **Cons:** Some PostgreSQL features unavailable

```bash
export DB_HOST="aws-1-us-east-2.pooler.supabase.com"
export DB_PORT="6543"
export DB_NAME="postgres"
export DB_USER="postgres.wxinjyotnrqxrwqrtvkp"
export DB_PASSWORD="jovpeW-pukgu0-nifron"
```

3. Connection Pooler - Session Mode (Port 5432)

Best for: Production with full PostgreSQL features **Pros:** Connection pooling + all PostgreSQL features **Cons:** Fewer concurrent connections than transaction mode

```bash
export DB_HOST="aws-1-us-east-2.pooler.supabase.com"
export DB_PORT="5432"
export DB_NAME="postgres"
export DB_USER="postgres.wxinjyotnrqxrwqrtvkp"
export DB_PASSWORD="jovpeW-pukgu0-nifron"
```

Recommended Setup

For Development (Local)

Use **Direct Connection** for simplicity:

```bash
# Create .env file in backend directory
cat > backend/.env << EOF
DB_HOST=db.wxinjyotnrqxrwqrtvkp.supabase.co
DB_PORT=5432
DB_NAME=postgres
DB_USER=postgres
DB_PASSWORD=jovpeW-pukgu0-nifron
EOF
```

For Production (Render)

Use **Connection Pooler - Session Mode** for best balance:

1. Go to Render Dashboard → Your Service → Environment
2. Add these environment variables:

```
DB_HOST=aws-1-us-east-2.pooler.supabase.com
DB_PORT=5432
DB_NAME=postgres
DB_USER=postgres.wxinjyotnrqxrwqrtvkp
DB_PASSWORD=jovpeW-pukgu0-nifron
```

Troubleshooting Connection Issues

Issue: "server closed the connection unexpectedly"

Causes:

1. Supabase free tier connection limit reached (check Supabase dashboard)
2. Idle connection timeout
3. Network issues between Render and Supabase

Solutions:

1. **Check Supabase Connection Limits:**
   - Go to Supabase Dashboard → Settings → Database

- Check "Connection limit" usage
- Free tier: 60 connections max

2. **Reduce Connection Pool Size:**

```python
# In the improved app.py
pool = ConnectionPool(
    CONNECTION_STRING,
    min_size=1,  # Reduce from 2
    max_size=5,  # Reduce from 10
    ...
)
```

3. **Enable Connection Pooling in Supabase:**
   - Always use the pooler URL for production
   - Transaction mode: Port 6543
   - Session mode: Port 5432 (recommended)

4. **Check Health Endpoint:**

```bash
curl https://jazzreference.onrender.com/api/health
```

Look for:

- `pool_available`: Should be > 0
- `requests_waiting`: Should be 0
- `db_version`: Should show PostgreSQL version

Issue: Intermittent Connection Failures

Solution: The improved app includes:

- Automatic retry logic (3 attempts)
- Connection keepalive settings
- Query timeouts
- Pool health monitoring

Issue: Running Out of Connections

Check Current Connections:

```sql
-- Run in Supabase SQL Editor
SELECT count(*) as connection_count
FROM pg_stat_activity
WHERE datname = 'postgres';
```

Fix:

1. Reduce `max_size` in connection pool
2. Use connection pooler (pooler URL)
3. Upgrade Supabase plan for more connections

Testing Your Configuration

1. Test Local Connection

```bash
cd backend
source venv/bin/activate
python app.py
```

2. Test Health Endpoint

```bash
# Local
curl http://localhost:5001/api/health | jq

# Production
curl https://jazzreference.onrender.com/api/health | jq
```

Expected output:

```json
```

```
{
  "status": "healthy",
  "database": "connected",
  "pool_stats": {
    "pool_size": 2,
    "pool_available": 2,
    "requests_waiting": 0
  },
  "db_version": "PostgreSQL 15.x...",
  "db_time": "2025-01-10 12:34:56.789"
}
```


3. Load Test (Optional)

```
bash

# Install apache bench if needed: brew install httpd (macOS)
ab -n 100 -c 10 http://localhost:5001/api/health
```

Look for: 0 failed requests

Monitoring

Watch Application Logs

```
bash

# In Render dashboard, go to Logs tab
# Look for these key messages:

✓ Connection pool initialized successfully
✓ Query executed in 0.023s
✗ Connection pool initialization failed
```

Key Metrics to Monitor

1. **Pool Availability:** Should stay > 0
2. **Query Duration:** Should be < 1 second
3. **Failed Requests:** Should be 0
4. **Requests Waiting:** Should be 0

Quick Fix Commands

Reset Everything

```bash
# Kill all Python processes
pkill -9 python

# Restart Flask
cd backend
source venv/bin/activate
python app.py
```

Force New Deployment on Render

```bash
# Trigger manual deploy from Render dashboard
# OR push empty commit to trigger deploy
git commit --allow-empty -m "Force redeploy"
git push
```

Emergency: Switch Connection Mode

```bash
# Update Render environment variables immediately:
# 1. Go to Render → Environment
# 2. Change DB_HOST and DB_PORT
# 3. Click "Save Changes"
# App will auto-restart with new settings
```