Contents

SSAT Q-Dashboard 개발 가이드

React + TypeScript + Supabase 풀스택 애플리케이션 구축

목차

---

1. 프로젝트 개요

1.1 프로젝트 목적

SSAT Q-Dashboard 는 제조업체의 품질 관리를 위한 종합 대시보드입니다.

1.2 주요 기능

- **NCR 관리** - 부적합 보고서 추적 및 관리
- **고객 품질 지표** - 고객별 월간 품질 성과 모니터링
- **협력업체 품질 지표** - 수입검사 결과 추적
- **출하 품질 지표** - 출하 품질 성과 모니터링
- **신속 대응 추적** - 불량 신속 대응 관리
- **공정 품질 대시보드** - Excel 데이터 업로드 및 분석

1.3 프로젝트 정보

- **버전**: 2.0.0
- **총 커밋 수**: 91 개
- **개발 기간**: 약 2 일 (집중 개발)
- **주요 컴포넌트**: 10 개

---

2. 기술 스택 선정

2.1 프론트엔드

React 18.3.1

- **선택 이유**: 컴포넌트 기반 아키텍처, 풍부한 생태계
- **장점**: 재사용 가능한 UI 컴포넌트, Virtual DOM 성능 최적화

TypeScript 5.7.2

- **선택 이유**: 타입 안정성, IDE 지원 강화
- **장점**: 컴파일 타임 에러 검출, 유지보수성 향상

Vite 6.0.5

- **선택 이유**: 빠른 빌드 속도, 개발 서버 Hot Module Replacement
- **장점**: ES 모듈 기반, 번들 크기 최적화

Tailwind CSS 3.4.17

- **선택 이유**: 유틸리티 우선 CSS 프레임워크
- **장점**: 빠른 스타일링, 일관된 디자인 시스템

2.2 백엔드 & 데이터베이스

Supabase (PostgreSQL)

- **선택 이유**:
  - 오픈소스 Firebase 대안
  - PostgreSQL 기반 (강력한 관계형 DB)
  - 실시간 기능 내장
  - RESTful API 자동 생성
- **장점**:
  - 무료 티어 제공
  - 별도 서버 불필요
  - Row Level Security (RLS) 지원

2.3 라이브러리

라이브러리	버전	용도
`@supabase/supabase-js`	2.45.0	Supabase 클라이언트
`recharts`	2.12.7	차트 시각화
`xlsx`	0.18.5	Excel 파일 처리
`jspdf`	2.5.1	PDF 생성
`html2canvas`	1.4.1	HTML to 이미지 변환
`@google/generative-ai`	0.21.0	AI 기능 (Gemini)

3. 개발 환경 구축

3.1 프로젝트 초기화

```
# Vite 로 React + TypeScript 프로젝트 생성
npm create vite@latest ssat-q-dashboard -- --template react-ts

# 프로젝트 디렉토리로 이동
cd ssat-q-dashboard
```

```
# 의존성 설치
npm install
```

3.2 필수 패키지 설치

```
# Supabase 클라이언트
npm install @supabase/supabase-js

# UI 라이브러리
npm install recharts xlsx jspdf html2canvas

# Tailwind CSS
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

3.3 Tailwind CSS 설정

tailwind.config.js:

```
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

3.4 Vite 설정

vite.config.ts:

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

export default defineConfig({
  plugins: [react()],
  build: {
    outDir: 'dist',
    minify: 'esbuild', // terser 대신 esbuild 사용
  }
})
```

3.5 프로젝트 구조

```
ssat-q-dashboard/
   components/              # React 컴포넌트
      Dashboard.tsx         # 메인 대시보드
      NCRForm.tsx           # NCR 입력 폼
      NCRTable.tsx          # NCR 테이블
      CustomerQuality.tsx   # 고객 품질
      IncomingQuality.tsx   # 수입검사
      OutgoingQuality.tsx   # 출하품질
      QuickResponse.tsx     # 신속대응
      ProcessQuality.tsx    # 공정품질
```

```
    DatabaseSetupScreen.tsx  # DB 설정 화면
  lib/
    supabaseClient.ts    # Supabase 클라이언트
    dbMigration.ts       # DB 마이그레이션
  types.ts               # TypeScript 타입 정의
  App.tsx                # 메인 앱 컴포넌트
  index.tsx              # 엔트리 포인트
  supabase-schema.sql    # DB 스키마
  package.json           # 프로젝트 설정
```

---

4. 데이터베이스 설계

4.1 Supabase 프로젝트 생성

1. **Supabase 회원가입**
   - https://supabase.com 접속
   - GitHub 또는 이메일로 가입
2. **새 프로젝트 생성**
   - 프로젝트 이름: `ssat-q-dashboard`
   - 데이터베이스 비밀번호 설정
   - 리전 선택 (권장: Northeast Asia - Seoul)
3. **프로젝트 정보 확인**
   - Project URL: `https://xxxxx.supabase.co`
   - API Keys → `anon public` 키 복사

4.2 데이터베이스 스키마 설계

테이블 구조 **1. NCR Entries (ncr_entries)**

```sql
CREATE TABLE ncr_entries (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  month INTEGER NOT NULL,
  day INTEGER NOT NULL,
  source TEXT NOT NULL,
  customer TEXT NOT NULL,
  model TEXT NOT NULL,
  part_name TEXT NOT NULL,
  part_no TEXT NOT NULL,
  defect_content TEXT,
  outflow_cause TEXT,
  root_cause TEXT,
  countermeasure TEXT,
  plan_date TEXT,
  result_date TEXT,
  effectiveness_check TEXT,
  status TEXT NOT NULL DEFAULT 'Open',
  progress_rate INTEGER DEFAULT 0,
  remarks TEXT,
  attachments JSONB DEFAULT '[]'::jsonb,
  eight_d_data JSONB,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

2. Customer Metrics (customer_metrics)

```sql
CREATE TABLE customer_metrics (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  year INTEGER NOT NULL,
  customer TEXT NOT NULL,
  month INTEGER NOT NULL CHECK (month >= 1 AND month <= 12),
  target NUMERIC NOT NULL DEFAULT 10,
  inspection_qty INTEGER NOT NULL DEFAULT 0,
  defects INTEGER NOT NULL DEFAULT 0,
  actual NUMERIC NOT NULL DEFAULT 0,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  CONSTRAINT unique_customer_year_month UNIQUE (customer, year, month)
);
```

3. Supplier Metrics (supplier_metrics)

```sql
CREATE TABLE supplier_metrics (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  year INTEGER NOT NULL,
  supplier TEXT NOT NULL,
  month INTEGER NOT NULL CHECK (month >= 1 AND month <= 12),
  target NUMERIC NOT NULL DEFAULT 7500,
  incoming_qty INTEGER NOT NULL DEFAULT 0,
  inspection_qty INTEGER NOT NULL DEFAULT 0,
  defects INTEGER NOT NULL DEFAULT 0,
  actual NUMERIC NOT NULL DEFAULT 0,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  CONSTRAINT unique_supplier_year_month UNIQUE (supplier, year, month)
);
```

4. Outgoing Metrics (outgoing_metrics)

```sql
CREATE TABLE outgoing_metrics (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  year INTEGER NOT NULL,
  month INTEGER NOT NULL CHECK (month >= 1 AND month <= 12),
  target NUMERIC NOT NULL DEFAULT 10,
  inspection_qty INTEGER NOT NULL DEFAULT 0,
  defects INTEGER NOT NULL DEFAULT 0,
  actual NUMERIC NOT NULL DEFAULT 0,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  CONSTRAINT unique_outgoing_year_month UNIQUE (year, month)
);
```

5. Quick Response Entries (quick_response_entries)

```sql
CREATE TABLE quick_response_entries (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  date DATE NOT NULL,
  department TEXT NOT NULL,
  machine_no TEXT,
  defect_count INTEGER NOT NULL DEFAULT 0,
  model TEXT NOT NULL,
  defect_type TEXT,
  process TEXT,
  defect_content TEXT,
```

```
  coating TEXT,
  area TEXT,
  material_code TEXT,
  shielding TEXT,
  action TEXT,
  material_manager TEXT,
  meeting_attendance TEXT,
  status_24h TEXT DEFAULT 'N/A',
  status_3d TEXT DEFAULT 'N/A',
  status_14day TEXT DEFAULT 'N/A',
  status_24d TEXT DEFAULT 'N/A',
  status_25d TEXT DEFAULT 'N/A',
  status_30d TEXT DEFAULT 'N/A',
  customer_mm TEXT,
  remarks TEXT,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

6. Process Quality Uploads (process_quality_uploads)

```
CREATE TABLE process_quality_uploads (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  filename TEXT NOT NULL,
  record_count INTEGER NOT NULL DEFAULT 0,
  upload_date TIMESTAMPTZ DEFAULT NOW(),
  created_at TIMESTAMPTZ DEFAULT NOW()
);
```

7. Process Quality Data (process_quality_data)

```
CREATE TABLE process_quality_data (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  upload_id UUID REFERENCES process_quality_uploads(id) ON DELETE CASCADE,
  customer TEXT NOT NULL,
  part_type TEXT NOT NULL,
  vehicle_model TEXT,
  product_name TEXT,
  production_qty INTEGER NOT NULL DEFAULT 0,
  defect_qty INTEGER NOT NULL DEFAULT 0,
  defect_amount NUMERIC NOT NULL DEFAULT 0,
  defect_rate NUMERIC NOT NULL DEFAULT 0,
  data_date DATE NOT NULL,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

4.3 트리거 및 인덱스

```
-- 트리거 함수
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
   NEW.updated_at = NOW();
   RETURN NEW;
END;
$$ language 'plpgsql';
```

```sql
-- 각 테이블에 트리거 적용
CREATE TRIGGER update_ncr_entries_updated_at
  BEFORE INSERT OR UPDATE ON ncr_entries
  FOR EACH ROW
  EXECUTE FUNCTION update_updated_at_column();

-- (다른 테이블들도 동일하게 적용)
```

Updated_at 자동 업데이트 트리거

```sql

-- 검색 성능 향상을 위한 인덱스
CREATE INDEX idx_ncr_customer ON ncr_entries(customer);
CREATE INDEX idx_ncr_status ON ncr_entries(status);
CREATE INDEX idx_ncr_created ON ncr_entries(created_at DESC);
CREATE INDEX idx_customer_metrics_year ON customer_metrics(year);
CREATE INDEX idx_pq_data_date ON process_quality_data(data_date DESC);
```

인덱스 생성

4.4 스키마 적용 방법

1. **Supabase Dashboard → SQL Editor**
2. **New Query** 클릭
3. **supabase-schema.sql** 내용 복사 & 붙여넣기
4. **Run** 버튼 클릭 (**Ctrl/Cmd + Enter**)
5. **Table Editor** 에서 테이블 생성 확인

---

5. 프론트엔드 개발

5.1 타입 정의 (types.ts)

```typescript
// NCR Entry 타입
export interface NCREntry {
  id: string;
  month: number;
  day: number;
  source: string;
  customer: string;
  model: string;
  part_name: string;
  part_no: string;
  defect_content?: string;
  status: 'Open' | 'In Progress' | 'Closed';
  progress_rate: number;
  created_at: string;
  updated_at: string;
}

// Customer Metric 타입
export interface CustomerMetric {
  id: string;
  year: number;
  customer: string;
  month: number;
```

```
  target: number;
  inspection_qty: number;
  defects: number;
  actual: number;
}

// Process Quality Data 타입
export interface ProcessQualityData {
  id: string;
  customer: string;
  part_type: string;
  vehicle_model?: string;
  product_name?: string;
  production_qty: number;
  defect_qty: number;
  defect_amount: number;
  defect_rate: number;
  data_date: string;
}
```

5.2 주요 컴포넌트 구조

```
import React, { useState } from 'react';
import NCRTable from './components/NCRTable';
import CustomerQuality from './components/CustomerQuality';
import IncomingQuality from './components/IncomingQuality';
// ... 다른 컴포넌트 import

export default function Dashboard() {
  const [activeTab, setActiveTab] = useState('dashboard');

  return (
    <div className="min-h-screen bg-gray-50">
      {/* 네비게이션 바 */}
      <nav className="bg-white shadow">
        <div className="flex space-x-4">
          <button onClick={() => setActiveTab('dashboard')}>
            대시보드
          </button>
          <button onClick={() => setActiveTab('customer')}>
            고객 품질
          </button>
          {/* ... 다른 탭들 */}
        </div>
      </nav>

      {/* 콘텐츠 영역 */}
      <main className="container mx-auto p-4">
        {activeTab === 'dashboard' && <NCRTable />}
        {activeTab === 'customer' && <CustomerQuality />}
        {/* ... 다른 탭 컴포넌트들 */}
      </main>
    </div>
  );
}
```

Dashboard.tsx (메인 대시보드)

```
import React, { useState, useEffect } from 'react';
import { supabase } from '../lib/supabaseClient';
import { BarChart, Bar, XAxis, YAxis, CartesianGrid } from 'recharts';

export default function CustomerQuality() {
  const [metrics, setMetrics] = useState<CustomerMetric[]>([]);
  const [year, setYear] = useState(new Date().getFullYear());

  useEffect(() => {
    loadMetrics();
  }, [year]);

  const loadMetrics = async () => {
    const { data, error } = await supabase
      .from('customer_metrics')
      .select('*')
      .eq('year', year)
      .order('month', { ascending: true });

    if (data) setMetrics(data);
  };

  return (
    <div className="space-y-4">
      {/* 연도 선택 */}
      <select value={year} onChange={(e) => setYear(Number(e.target.value))}>
        <option value={2024}>2024</option>
        <option value={2025}>2025</option>
        <option value={2026}>2026</option>
      </select>

      {/* 차트 */}
      <BarChart width={800} height={400} data={metrics}>
        <CartesianGrid strokeDasharray="3 3" />
        <XAxis dataKey="month" />
        <YAxis />
        <Bar dataKey="actual" fill="#3b82f6" />
      </BarChart>

      {/* 데이터 입력 폼 */}
      {/* ... */}
    </div>
  );
}
```

CustomerQuality.tsx (고객 품질 관리)

```
import React, { useState } from 'react';
import * as XLSX from 'xlsx';
import { supabase } from '../lib/supabaseClient';

export default function ProcessQuality() {
  const [uploading, setUploading] = useState(false);
```

```tsx
const handleFileUpload = async (e: React.ChangeEvent<HTMLInputElement>) => {
  const file = e.target.files?.[0];
  if (!file) return;

  setUploading(true);

  try {
    // 1. Excel 파일 읽기
    const data = await file.arrayBuffer();
    const workbook = XLSX.read(data);
    const worksheet = workbook.Sheets[workbook.SheetNames[0]];
    const jsonData = XLSX.utils.sheet_to_json(worksheet);

    // 2. 업로드 기록 생성
    const { data: uploadRecord, error: uploadError } = await supabase
      .from('process_quality_uploads')
      .insert({
        filename: file.name,
        record_count: jsonData.length
      })
      .select()
      .single();

    // 3. 데이터 변환 및 저장
    const processedData = jsonData.map((row: any) => ({
      upload_id: uploadRecord.id,
      customer: row['고객사'] || row['거래처'],
      part_type: row['부품유형'] || row['공정'],
      production_qty: Number(row['생산수량'] || row['생산량']),
      defect_qty: Number(row['불량수량'] || row['불량량']),
      defect_amount: Number(row['불량금액'] || row['금액']),
      defect_rate: (Number(row['불량수량']) / Number(row['생산수량'])) * 100,
      data_date: row['일자'] || row['날짜']
    }));

    const { error: dataError } = await supabase
      .from('process_quality_data')
      .insert(processedData);

    if (dataError) throw dataError;

    alert('업로드 성공!');
  } catch (error) {
    console.error('업로드 실패:', error);
    alert('업로드 실패');
  } finally {
    setUploading(false);
  }
};

return (
  <div>
    <input
      type="file"
      accept=".xlsx,.xls"
```

```
        onChange={handleFileUpload}
        disabled={uploading}
      />
      {uploading && <p>업로드 중...</p>}
    </div>
  );
}
```

ProcessQuality.tsx (공정 품질 - Excel 업로드)

---

6. 백엔드 연동

6.1 Supabase 클라이언트 설정

lib/supabaseClient.ts:

```
import { createClient } from '@supabase/supabase-js';

// 기본값 설정
const DEFAULT_URL = 'https://xjjsqyawvojybuyrehrr.supabase.co';
const DEFAULT_KEY = 'sb_publishable_vvaBbJHvo0pcUwaUiPW0ww_vElHC7GW';

// 저장소 키 (v5 - 기존 캐시 무효화)
const STORAGE_KEY_URL = 'supabase_url_v5';
const STORAGE_KEY_KEY = 'supabase_key_v5';

export const getSupabase = () => {
  let url = DEFAULT_URL;
  let key = DEFAULT_KEY;

  // LocalStorage 에서 사용자 정의 설정 읽기
  if (typeof window !== 'undefined') {
    const storedUrl = localStorage.getItem(STORAGE_KEY_URL);
    const storedKey = localStorage.getItem(STORAGE_KEY_KEY);

    if (storedUrl && storedUrl.trim().startsWith('http')) {
      url = storedUrl.trim();
    }
    if (storedKey && storedKey.trim().length > 0) {
      key = storedKey.trim();
    }
  }

  try {
    return createClient(url, key);
  } catch (error) {
    console.error("Supabase 초기화 실패:", error);
    return createClient(DEFAULT_URL, DEFAULT_KEY);
  }
};

export const supabase = getSupabase();

// 설정 저장 함수
export const saveSupabaseConfig = (newUrl: string, newKey: string) => {
  const cleanUrl = newUrl.trim();
```

```
  const cleanKey = newKey.trim();

  if (!cleanUrl.startsWith('http')) {
    alert('유효하지 않은 URL 입니다.');
    return;
  }

  localStorage.setItem(STORAGE_KEY_URL, cleanUrl);
  localStorage.setItem(STORAGE_KEY_KEY, cleanKey);

  // 새로고침하여 설정 적용
  window.location.href = '/';
};

// 설정 초기화 함수
export const resetSupabaseConfig = () => {
  localStorage.removeItem(STORAGE_KEY_URL);
  localStorage.removeItem(STORAGE_KEY_KEY);
  window.location.href = '/';
};
```

6.2 CRUD 작업 예제

```
// 전체 조회
const { data, error } = await supabase
  .from('ncr_entries')
  .select('*')
  .order('created_at', { ascending: false });

// 필터 조회
const { data } = await supabase
  .from('customer_metrics')
  .select('*')
  .eq('year', 2025)
  .gte('month', 1)
  .lte('month', 12);
```

데이터 조회 (Read)

```
const { data, error } = await supabase
  .from('ncr_entries')
  .insert({
    month: 1,
    day: 15,
    customer: 'Samsung',
    model: 'Model-X',
    part_name: 'Engine Part',
    part_no: 'EP-001',
    status: 'Open',
    progress_rate: 0
  })
  .select()
  .single();
```

데이터 생성 (Create)

```
const { error } = await supabase
  .from('ncr_entries')
  .update({
    status: 'Closed',
    progress_rate: 100
  })
  .eq('id', entryId);
```

데이터 수정 (Update)

```
const { error } = await supabase
  .from('ncr_entries')
  .delete()
  .eq('id', entryId);
```

데이터 삭제 (Delete)

```
const { error } = await supabase
  .from('customer_metrics')
  .upsert({
    customer: 'Samsung',
    year: 2025,
    month: 1,
    target: 10,
    inspection_qty: 1000,
    defects: 5,
    actual: 5
  }, {
    onConflict: 'customer,year,month' // UNIQUE 제약조건 컬럼
  });
```

Upsert (있으면 업데이트, 없으면 생성)

---

7. 배포 및 운영

7.1 Vercel 배포

```
# Git 초기화
git init
git add .
git commit -m "Initial commit"

# GitHub 저장소 생성 후 푸시
git remote add origin https://github.com/username/ssat-q-dashboard.git
git branch -M main
git push -u origin main
```

단계 1: GitHub 저장소 연결

단계 2: Vercel 배포

1. Vercel 회원가입 (https://vercel.com)

2. **Import Git Repository**
3. **프로젝트 선택**: ssat-q-dashboard
4. **Build 설정**:
   - Framework Preset: Vite
   - Build Command: `npm run build`
   - Output Directory: `dist`
5. **환경 변수** (선택사항):
   - 기본 설정을 사용하므로 불필요
6. **Deploy** 클릭

단계 3: 배포 확인

- 배포 URL: `https://ssat-q-dashboard.vercel.app`
- 자동 배포: main 브랜치에 푸시할 때마다 자동 배포

7.2 환경 설정

```
// lib/supabaseClient.ts
const isProduction = import.meta.env.PROD;

const DEFAULT_URL = isProduction
  ? 'https://production.supabase.co'
  : 'https://dev.supabase.co';
```

Production vs Development

```
# .env.local (Git 에 커밋하지 않음)
VITE_SUPABASE_URL=https://xxxxx.supabase.co
VITE_SUPABASE_ANON_KEY=your-anon-key
```

```
// 사용법
const supabaseUrl = import.meta.env.VITE_SUPABASE_URL;
const supabaseKey = import.meta.env.VITE_SUPABASE_ANON_KEY;
```

환경 변수 (.env)

7.3 성능 최적화

```
// vite.config.ts
export default defineConfig({
  build: {
    minify: 'esbuild', // terser 보다 빠름
    rollupOptions: {
      output: {
        manualChunks: {
          'react-vendor': ['react', 'react-dom'],
          'chart-vendor': ['recharts'],
          'supabase-vendor': ['@supabase/supabase-js']
        }
      }
    }
  }
});
```

빌드 최적화

```
// 지연 로딩
const ProcessQuality = React.lazy(() => import('./components/ProcessQuality'));

// 사용
<Suspense fallback={<Loading />}>
  <ProcessQuality />
</Suspense>
```

코드 스플리팅

---

8. 문제 해결 및 디버깅

8.1 주요 문제 및 해결

문제 1: "updated_at 컬럼을 찾을 수 없음" 증상:

```
record 'new' has no field 'updated_at'
Could not find the 'updated_at' column
```

원인: - 데이터베이스 트리거가 제대로 생성되지 않음 - 테이블에 updated_at 컬럼이 없음

해결책:

```
-- 1. 컬럼 추가
ALTER TABLE customer_metrics
ADD COLUMN IF NOT EXISTS updated_at TIMESTAMPTZ DEFAULT NOW();

-- 2. 트리거 재생성
DROP TRIGGER IF EXISTS update_customer_metrics_updated_at
ON customer_metrics;

CREATE TRIGGER update_customer_metrics_updated_at
  BEFORE INSERT OR UPDATE ON customer_metrics
  FOR EACH ROW
  EXECUTE FUNCTION update_updated_at_column();
```

해결 과정: 1. verify-triggers.html 파일 생성 (트리거 검증 도구) 2. fix-updated-at-triggers.sql 스크립트 작성 3. Supabase SQL Editor 에서 실행 4. 문제 해결 확인

문제 2: Excel 업로드 시 NaN 값 처리 증상:

```
NULL value in column violates not-null constraint
```

원인: - Excel 빈 셀이 NaN 으로 변환됨 - NOT NULL 제약조건 위반

해결책:

```
const processedData = jsonData.map((row: any) => ({
  production_qty: Number(row['생산수량']) || 0, // 기본값 0
  defect_qty: Number(row['불량수량']) || 0,
  defect_amount: Number(row['불량금액']) || 0,
  defect_rate: isNaN(defectRate) ? 0 : defectRate
}));
```

문제 3: Terser 빌드 실패 증상:

```
Error: Build failed due to minify error
```

원인: - Terser 메모리 부족 - 복잡한 코드 minify 실패

해결책:

```
// vite.config.ts
export default defineConfig({
  build: {
    minify: 'esbuild' // terser → esbuild 로 변경
  }
});
```

문제 4: 중복 데이터 저장 **증상:** - 같은 고객/년도/월 데이터가 중복 저장됨

원인: - UNIQUE 제약조건 없음

해결책:

```
-- UNIQUE 제약조건 추가
ALTER TABLE customer_metrics
ADD CONSTRAINT unique_customer_year_month
UNIQUE (customer, year, month);

-- Upsert 사용
await supabase
  .from('customer_metrics')
  .upsert(data, {
    onConflict: 'customer,year,month'
  });
```

8.2 디버깅 도구

```
<!DOCTYPE html>
<html>
<head>
  <title>Database Verification</title>
  <script src="https://cdn.jsdelivr.net/npm/@supabase/supabase-js@2"></script>
</head>
<body>
  <h1>Database Table Verification</h1>
  <input id="url" placeholder="Supabase URL" />
  <input id="key" placeholder="API Key" />
  <button onclick="verifyTables()">Verify Tables</button>
  <div id="result"></div>

  <script>
    async function verifyTables() {
      const url = document.getElementById('url').value;
      const key = document.getElementById('key').value;
      const supabase = window.supabase.createClient(url, key);

      const tables = [
        'ncr_entries',
        'customer_metrics',
        'supplier_metrics',
        'outgoing_metrics',
        'quick_response_entries',
        'process_quality_uploads',
        'process_quality_data'
```

```
      ];

      let html = '<h2>Results:</h2>';
      for (const table of tables) {
        const { data, error } = await supabase
          .from(table)
          .select('count', { count: 'exact', head: true });

        if (error) {
          html += `<p> ${table}: ${error.message}</p>`;
        } else {
          html += `<p> ${table}: EXISTS</p>`;
        }
      }

      document.getElementById('result').innerHTML = html;
    }
  </script>
</body>
</html>
```

1. 데이터베이스 검증 도구 **(verify-database.html)**

```
<!-- 트리거 존재 여부 확인 -->
<script>
async function checkTriggers() {
  const { data, error } = await supabase
    .rpc('get_triggers'); // Custom function 필요

  if (data) {
    console.log('Triggers:', data);
  }
}
</script>
```

2. 트리거 검증 도구 **(verify-triggers.html)**

```
// 디버그 모드
const DEBUG = true;

const loadData = async () => {
  if (DEBUG) console.log('Loading data...');

  const { data, error } = await supabase
    .from('ncr_entries')
    .select('*');

  if (DEBUG) {
    console.log('Data:', data);
    console.log('Error:', error);
  }
};
```

3. 브라우저 콘솔 디버깅

8.3 개발 과정 주요 커밋

```
e37fb75 - fix: Add comprehensive solution for customer_metrics updated_at column error
5f181d0 - fix: Resolve customer quality input saving issue
0941a79 - fix: Add tools and documentation to fix updated_at trigger errors
1080fa7 - style: Update dashboard to Apple-style minimalist design
15aaef2 - feat: Add automatic database setup screen with table detection
9491ab1 - feat: Integrate process quality dashboard with Excel upload
a7cd0df - fix: Change minify from terser to esbuild to fix build failures
0188a5f - docs: Add comprehensive database setup guide and verification tool
```

---

9. 추가 학습 자료

9.1 공식 문서

- **React**: https://react.dev
- **TypeScript**: https://www.typescriptlang.org
- **Vite**: https://vitejs.dev
- **Tailwind CSS**: https://tailwindcss.com
- **Supabase**: https://supabase.com/docs
- **Recharts**: https://recharts.org

9.2 추천 학습 경로

1. **JavaScript/TypeScript 기초**
2. **React Hooks (useState, useEffect)**
3. **Supabase CRUD 작업**
4. **Tailwind CSS 유틸리티**
5. **Excel 파일 처리 (XLSX)**
6. **차트 라이브러리 (Recharts)**

9.3 다음 단계

☐ 사용자 인증 추가 (Supabase Auth)
☐ Row Level Security (RLS) 활성화
☐ 실시간 업데이트 (Supabase Realtime)
☐ 이메일 알림 (Supabase Edge Functions)
☐ 다국어 지원 (i18n)
☐ 모바일 반응형 개선
☐ PWA (Progressive Web App) 구현

---

10. 결론

이 프로젝트를 통해 다음을 학습했습니다:

풀스택 애플리케이션 개발 - React + TypeScript 프론트엔드 - Supabase 백엔드 - PostgreSQL 데이터베이스

실전 문제 해결 - 데이터베이스 트리거 문제 - Excel 파일 처리 - 빌드 최적화 - 배포 및 운영

모던 개발 도구 - Vite (빠른 빌드) - Tailwind CSS (유틸리티 CSS) - Vercel (자동 배포)

프로젝트 통계: - 총 커밋: 91 개 - 개발 기간: 2 일 - 컴포넌트: 10 개 - 데이터베이스 테이블: 7 개

부록

A. 자주 사용하는 명령어

```
# 개발 서버 시작
npm run dev

# 프로덕션 빌드
npm run build

# 빌드 미리보기
npm run preview

# Git 커밋
git add .
git commit -m "feat: Add new feature"
git push origin main

# 의존성 업데이트
npm update

# 캐시 삭제
npm cache clean --force
```

B. 유용한 VSCode 확장

- **ES7+ React/Redux/React-Native snippets**
- **Tailwind CSS IntelliSense**
- **Prettier - Code formatter**
- **ESLint**
- **GitLens**
- **Thunder Client** (API 테스트)

C. 프로젝트 체크리스트

개발 시작 전

- ☐ Node.js 설치 (v16+)
- ☐ Git 설치
- ☐ VSCode 설치
- ☐ Supabase 계정 생성

개발 중

- ☐ 타입 정의 작성
- ☐ 컴포넌트 분리
- ☐ 에러 핸들링
- ☐ 로딩 상태 처리
- ☐ Git 커밋 규칙 준수

배포 전

- ☐ 빌드 테스트
- ☐ 환경 변수 확인
- ☐ 데이터베이스 스키마 적용
- ☐ CORS 설정 확인
- ☐ 성능 테스트

문서 작성일: 2026 년 1 월 19 일 **버전**: 2.0.0 **작성자**: SSAT Q-Dashboard Team

---

이 문서는 교육용으로 작성되었습니다. 실제 프로젝트에 맞게 수정하여 사용하시기 바랍니다.