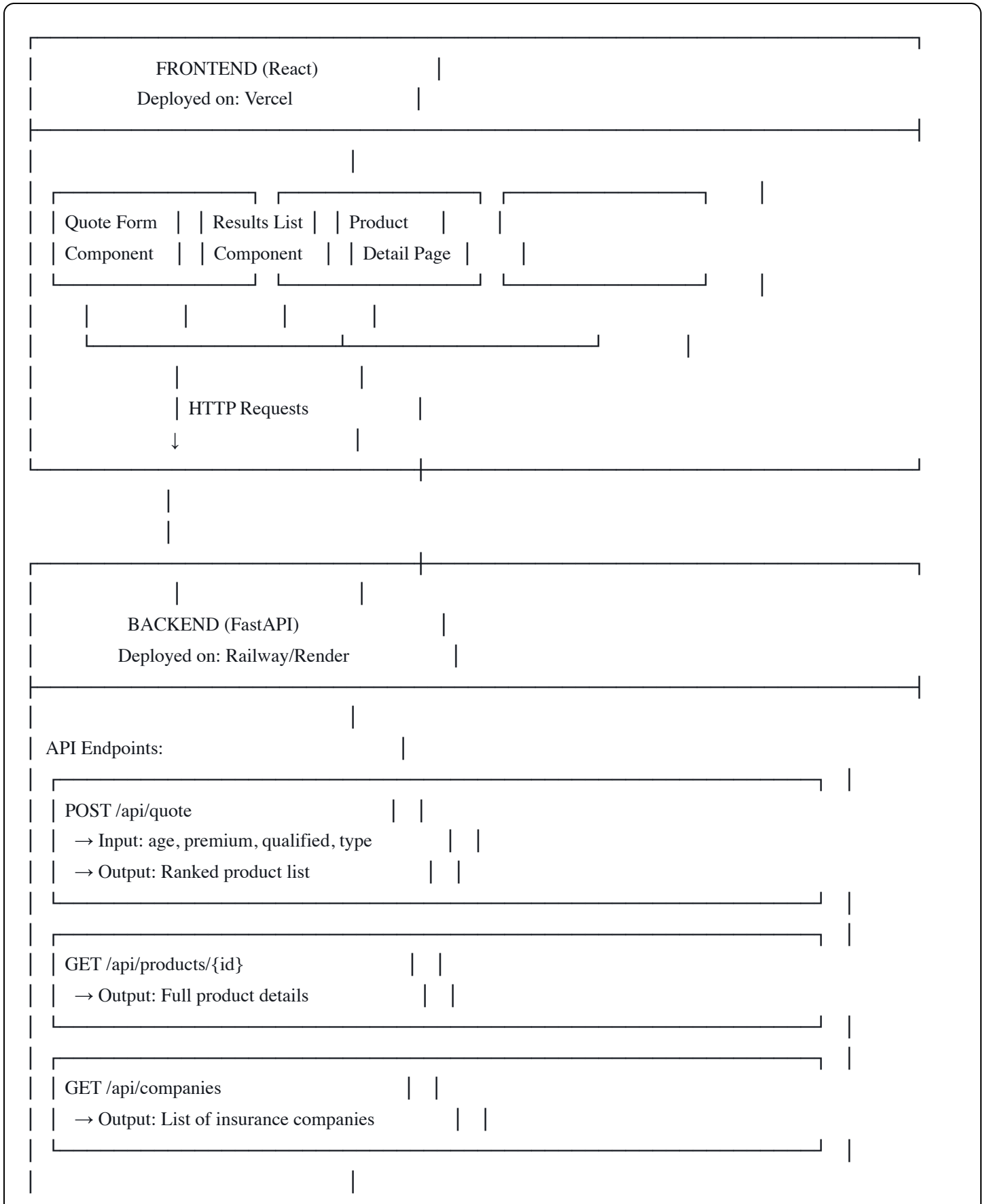
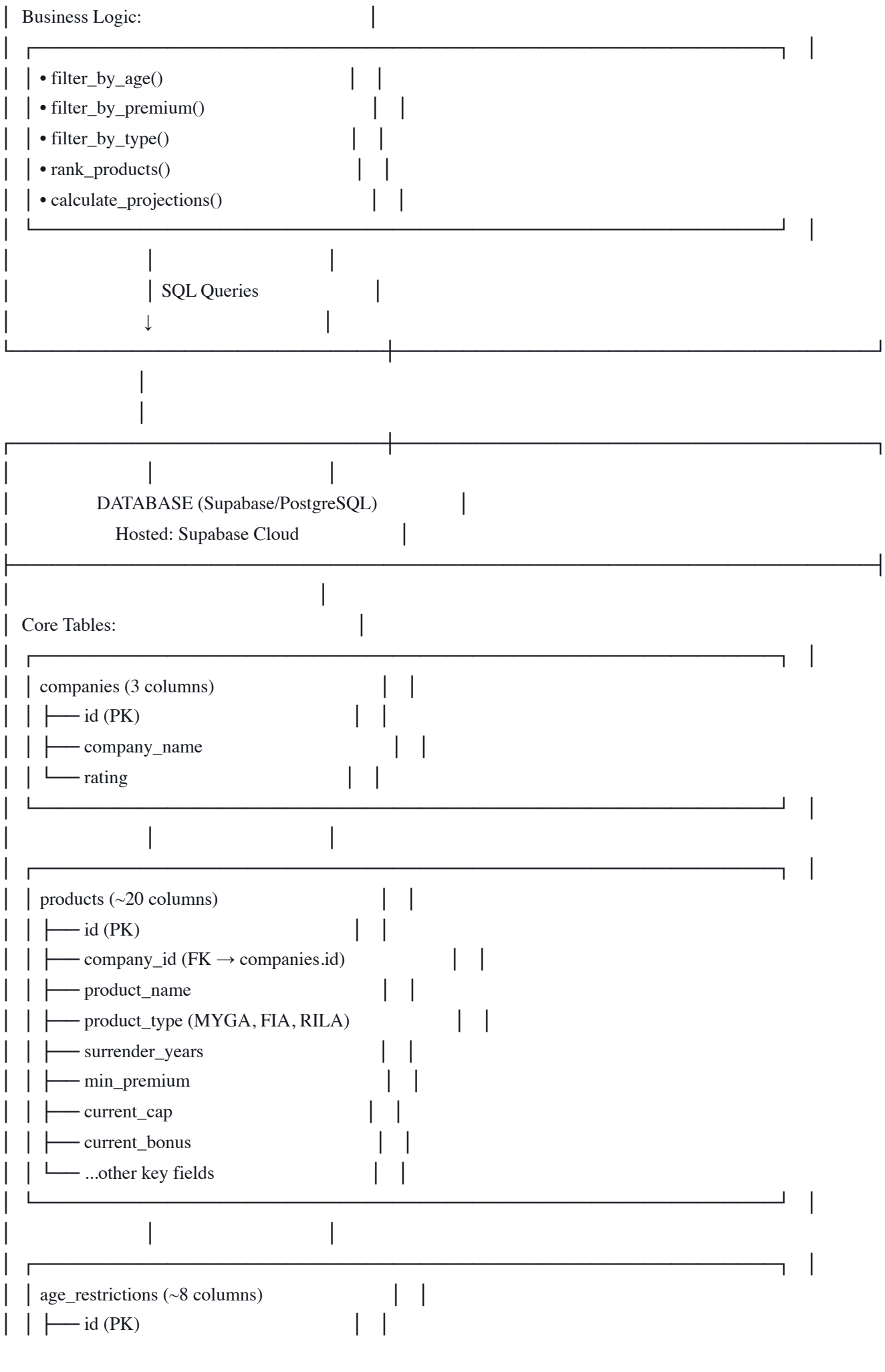
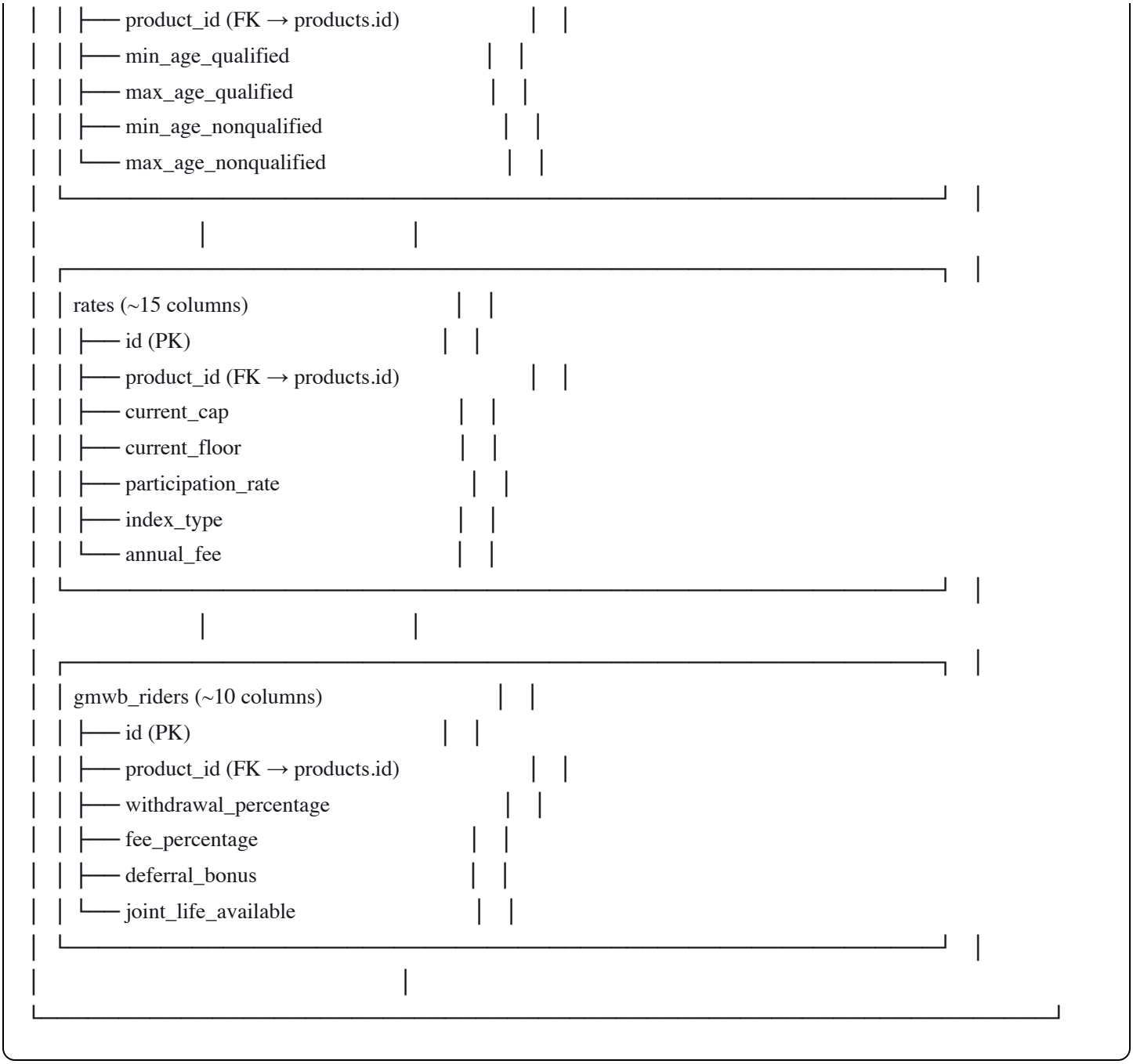
System Architecture - Quick Reference

🏛️ Three-Tier Architecture




Business Logic:
• filter_by_age()
• filter_by_premium()
• filter_by_type()
• rank_products()
• calculate_projections()
SQL Queries
DATABASE (Supabase/PostgreSQL)
Hosted: Supabase Cloud
Core Tables:
companies (3 columns)
├── id (PK)
├── company_name
└── rating
products (~20 columns)
├── id (PK)
├── company_id (FK → companies.id)
├── product_name
├── product_type (MYGA, FIA, RILA)
├── surrender_years
├── min_premium
├── current_cap
├── current_bonus
└── ...other key fields
age_restrictions (~8 columns)
├── id (PK)



🔄 Request Flow Example

USER ACTION:

Insurance agent fills out form:

Age: 62

Premium: $100,000

Type: Non-Qualified

Product: MYGA

↓

FRONTEND (React):
1. Validate inputs
2. Create JSON payload
3. POST to /api/quote

```
        |
        ↓
```

BACKEND (FastAPI):
1. Receive request
2. Query database:

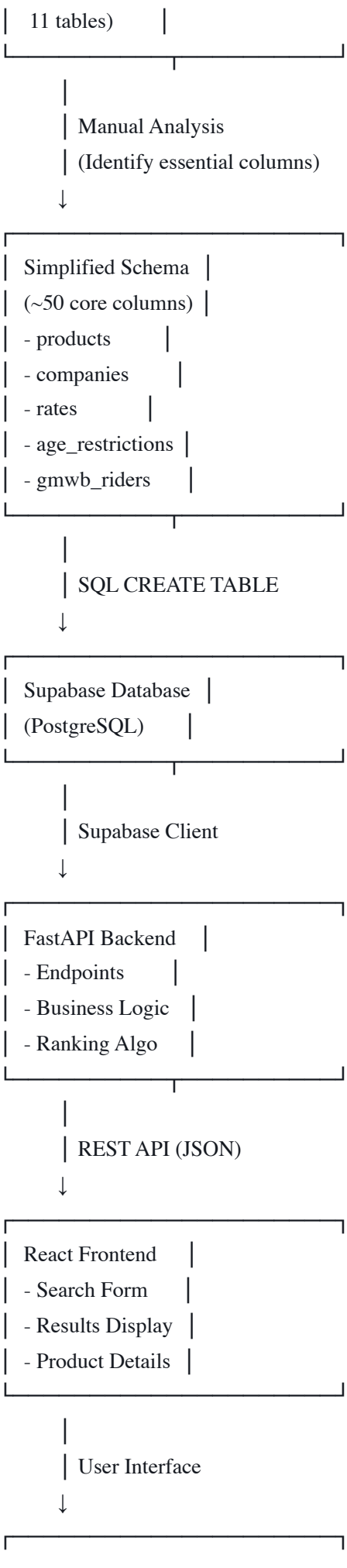
```
SELECT p.*, c.company_name, a.*, r.*
FROM products p
JOIN companies c ON p.company_id = c.id
JOIN age_restrictions a ON p.id = a.product_id
JOIN rates r ON p.id = r.product_id
WHERE a.min_age_nonqualified <= 62
  AND a.max_age_nonqualified >= 62
  AND p.product_type = 'MYGA'
  AND p.min_premium <= 100000
```

3. Rank results by current_cap DESC
4. Calculate projections
5. Return JSON response

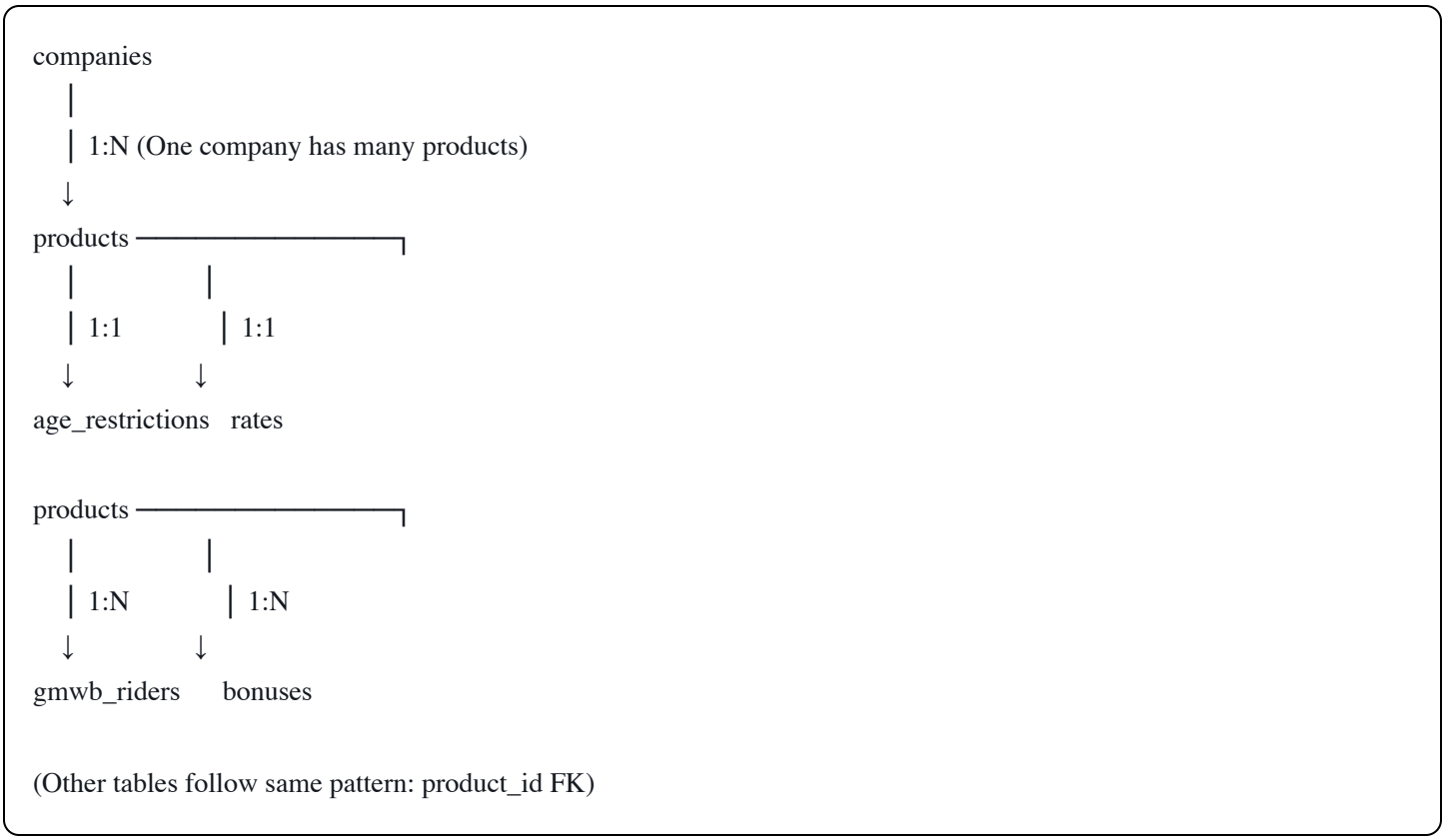
```
        |
        ↓
```

FRONTEND (React):
1. Receive product list
2. Display in table/cards
3. Show top 10 matches
4. Enable comparison/details actions

📊 Data Flow Diagram


11 tables)
Manual Analysis
(Identify essential columns)
Simplified Schema
(~50 core columns)
- products
- companies
- rates
- age_restrictions
- gmwb_riders
SQL CREATE TABLE
Supabase Database
(PostgreSQL)
Supabase Client
FastAPI Backend
- Endpoints
- Business Logic
- Ranking Algo
REST API (JSON)
React Frontend
- Search Form
- Results Display
- Product Details
User Interface

🎯 Core Data Relationships



🚀 Tech Stack Decision Matrix

Layer	Technology	Why Chosen
Frontend	React	- Component-based UI - Rich ecosystem - Easy state management - Fast development
Backend	FastAPI	- Python (great for data) - Fast performance - Auto API docs - Type validation

Layer	Technology	Why Chosen
Database	Supabase (PostgreSQL)	- Managed PostgreSQL - Real-time features - Built-in auth - Free tier
Hosting	Vercel + Railway	- Easy deployment - Auto HTTPS - CI/CD built-in - Free tiers

---

🔑 Key Design Patterns

1. Repository Pattern (Backend)

python

```python
class ProductRepository:
    def find_by_criteria(self, age, premium, qualified, type):
        # Encapsulate database queries
        pass

    def get_by_id(self, product_id):
        pass
```

2. Service Layer (Backend)

python

```python
class QuoteService:
    def __init__(self, product_repo):
        self.product_repo = product_repo

    def get_quote(self, criteria):
        products = self.product_repo.find_by_criteria(criteria)
        return self.rank_and_score(products)
```

3. Component Composition (Frontend)

jsx

```
<QuotePage>
  <QuoteForm onSubmit={handleSearch} />
  <ProductList products={results}>
    {results.map(p => (
      <ProductCard key={p.id} product={p} />
    ))}
  </ProductList>
</QuotePage>
```

Simplified Schema (MVP)

sql

```sql
-- Minimum viable schema (5 tables, ~50 columns total)

CREATE TABLE companies (
  id BIGSERIAL PRIMARY KEY,
  company_name TEXT NOT NULL,
  rating TEXT,
  created_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE products (
  id BIGSERIAL PRIMARY KEY,
  company_id BIGINT REFERENCES companies(id),
  product_name TEXT NOT NULL,
  product_type TEXT NOT NULL, -- MYGA, FIA, RILA
  surrender_years INT,
  min_premium NUMERIC,
  max_premium NUMERIC,
  current_cap NUMERIC,
  current_floor NUMERIC,
  participation_rate NUMERIC,
  current_bonus_rate NUMERIC,
  annual_fee NUMERIC,
  state_availability TEXT[], -- Array of state codes
  status TEXT DEFAULT 'active',
  created_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE age_restrictions (
  id BIGSERIAL PRIMARY KEY,
  product_id BIGINT REFERENCES products(id),
  min_age_qualified INT,
  max_age_qualified INT,
  min_age_nonqualified INT,
  max_age_nonqualified INT,
  max_annuitization_age INT
);

CREATE TABLE gmwb_riders (
  id BIGSERIAL PRIMARY KEY,
  product_id BIGINT REFERENCES products(id),
  rider_name TEXT,
  withdrawal_percentage NUMERIC,
  fee_percentage NUMERIC,
```

```
  deferral_bonus NUMERIC,
  roll_up_rate NUMERIC,
  joint_life_available BOOLEAN,
  available_start_age INT,
  available_end_age INT
);

CREATE TABLE surrender_charges (
  id BIGSERIAL PRIMARY KEY,
  product_id BIGINT REFERENCES products(id),
  year INT,
  charge_percentage NUMERIC
);

-- Indexes for performance
CREATE INDEX idx_products_type ON products(product_type);
CREATE INDEX idx_products_company ON products(company_id);
CREATE INDEX idx_age_restrictions_product ON age_restrictions(product_id);
CREATE INDEX idx_gmwb_product ON gmwb_riders(product_id);
```

🎨 UI Component Tree

```
App
├── Header
│   ├── Logo
│   └── Navigation
│
├── QuotePage (Route: /)
│   ├── QuoteForm
│   │   ├── AgeInput
│   │   ├── PremiumInput
│   │   ├── AccountTypeSelector
│   │   ├── ProductTypeSelector
│   │   └── AdvancedFilters (collapsible)
│   │
│   └── ProductList
│       ├── ProductCard (repeated)
│       │   ├── ProductHeader
│       │   ├── KeySpecs
│       │   └── ActionButtons
│       └── Pagination
```

```
│
├── ProductDetailPage (Route: /products/:id)
│   ├── ProductOverview
│   ├── SpecificationsTable
│   ├── GrowthChart
│   ├── IncomeProjections
│   └── SurrenderSchedule
│
└── ComparePage (Route: /compare)
    └── ComparisonTable
        ├── ProductColumn (2-4 products)
        └── SpecRows
```

⚡ Quick Command Reference

Supabase Setup

bash

```
# Install Supabase CLI
npm install -g supabase

# Login
supabase login

# Create project (or use web dashboard)
supabase projects create annuities-platform

# Run migrations
supabase db push
```

FastAPI Backend

bash

```
# Install dependencies
pip install fastapi uvicorn supabase python-dotenv

# Run development server
uvicorn main:app --reload --port 8000

# Run with auto-reload
uvicorn main:app --reload --host 0.0.0.0
```

React Frontend

```bash
# Create app
npx create-react-app annuities-frontend
cd annuities-frontend

# Install dependencies
npm install axios react-router-dom recharts

# Run dev server
npm start

# Build for production
npm run build
```

Deployment

```bash
# Frontend (Vercel)
npm install -g vercel
vercel login
vercel --prod

# Backend (Railway)
# 1. Connect GitHub repo
# 2. Railway auto-detects FastAPI
# 3. Add environment variables
# 4. Deploy

# Or use Render
# Similar process, connect repo
```

🎯 MVP Checklist

Database ✅

- [ ] Supabase project created
- [ ] 5 core tables created
- [ ] Sample data loaded (5-10 products)
- [ ] Foreign key relationships working
- [ ] Indexes added

Backend ✅

- [ ] FastAPI project setup
- [ ] Supabase client connected
- [ ] POST /api/quote endpoint working
- [ ] GET /api/products/:id endpoint working
- [ ] Filtering logic implemented
- [ ] Ranking algorithm working
- [ ] CORS configured
- [ ] Environment variables set

Frontend ✅

- [ ] React app created
- [ ] Quote form component
- [ ] Product list component
- [ ] Product card component
- [ ] API integration working
- [ ] Basic styling (Tailwind/CSS)
- [ ] Responsive design

Deployment ✅

- [ ] Frontend deployed (Vercel)
- [ ] Backend deployed (Railway/Render)
- [ ] Database connected
- [ ] Environment variables configured
- [ ] App is live and functional

Documentation ✅

- [ ] Process document written
- [ ] Architecture decisions explained
- [ ] AI usage documented
- [ ] Next steps outlined

🏆 Evaluation Rubric

Criteria	Weight	What They're Looking For
Data Understanding	30%	- Identified right columns - Understands relationships - Can explain table purposes
Architecture	30%	- Logical schema design - Sensible API structure - Scalable approach
Functionality	25%	- App works end-to-end - Results are accurate - Filters work correctly
Process & Docs	15%	- Clear thinking - Good trade-offs - Smart use of AI
BONUS: Innovation	+10%	- NLP query working - Unique features - Better than reference

---

This architecture document provides a complete blueprint for building your annuities quoting platform. Follow the layers from bottom (database) to top (UI) for a systematic implementation approach.