Build Your Own

Web Database Application

AI Prompt Templates & Step-by-Step Guide

A complete collection of prompts to build a professional web-database application using AI assistants like Claude, ChatGPT, or GitHub Copilot

How to Use This Guide

Follow the steps in order. Each step contains one or more prompt templates you can copy, customize for your own business scenario, and paste into an AI assistant. The AI will generate the code — your job is to review it, test it, and understand what it does.

Replace everything in [brackets] with your own details.

GMBA 612: Database Design & SQL — Spring 2026

The 10-Step Build Process

This guide follows the exact process used to build the KathyJane Skincare Store. Each step builds on the last. By the end, you'll have a complete, deployed web-database application.

Step	What You Build	What You Get
1	Define Your Business Concept	A clear scenario: what the store sells, who the customers are, what data you need
2	Design the Database Schema	An ERD and a SQL script that creates all tables with sample data
3	Generate the Setup Script	A PHP script that creates your entire site structure in one click
4	Build Customer-Facing Pages	Homepage, product listing, product detail, about page, newsletter signup
5	Build the Admin Panel	Login system, dashboard with stats, product editor, supplier report
6	Add the Shopping Cart	Session-based cart, checkout with simulated payments, order tracking
7	Add an Analytics Dashboard	API calls to government data, MySQL caching, Chart.js visualizations
8	Create Brand Assets	Hero images, product photos, a Best Sellers spotlight page
9	Style and Polish	Custom CSS matching your brand, Google Fonts, responsive design
10	Deploy and Test	Upload to Hostinger, configure database, test everything live

i AI Collaboration Rule: Always review AI-generated code before running it. Ask yourself: What did it assume? What would break this? Would I bet my grade on this? The AI is your collaborator, not your replacement.

Step 1: Define Your Business Concept

Before writing any code, clearly define what you're building. The AI works best when you give it specific context about your business scenario.

Prompt 1A: Business Concept Brief

```
I'm building a web-database application for a fictional business called
[YOUR BUSINESS NAME].

Business type: [e.g., online skincare store, pet grooming salon, fitness
studio, bookstore]
Target customers: [e.g., women aged 25-45, pet owners, college students]
Products/services: [list 5-10 items you want in the database]

The store needs:
- A customer-facing website where people can browse [products/services]
- An admin panel where the owner can manage [products/services] and view
reports
- A MySQL database hosted on shared web hosting (Hostinger)
- Clean, professional design with a [describe your aesthetic: warm,
modern, playful, luxury]

Please help me define:
1. The core entities (tables) I'll need
2. A sample list of 8-10 products/services with realistic names,
descriptions, and prices
3. The types of reports the admin would want to see
4. A one-paragraph "About" story for the fictional brand
```

→ This gives the AI everything it needs to create consistent content throughout the entire project.

U0001F4A1 Pro Tip: Pick a business you find interesting — you'll be working with this data for weeks. A bakery, a music store, a yoga studio, a sneaker shop, a pet adoption agency — anything with products or services and customers.

Step 2: Design the Database Schema

Now you'll ask the AI to create your database structure. This prompt generates the SQL script you'll import into phpMyAdmin.

Prompt 2A: SQL Database Script

```
Create a complete MySQL SQL script for my [YOUR BUSINESS NAME] web
application.

The database should include these tables:
- categories (for organizing [products/services])
- suppliers (companies that supply our [products])
- [products/services] (the main items we sell, linked to categories and
suppliers)
- newsletter_subscribers (email signups)
- admin_users (login credentials for the admin panel)

Requirements:
- Use InnoDB engine and utf8mb4 charset
- Include proper PRIMARY KEYs (AUTO_INCREMENT)
- Include FOREIGN KEYs linking [products] to categories and suppliers
- Add created_at DATETIME DEFAULT CURRENT_TIMESTAMP columns
- Include 5 sample categories, 4 sample suppliers, and 10 sample
[products]
- Include 1 default admin user (username: admin, password hash for
"admin2025" using bcrypt)
- Include 3 sample newsletter subscribers
- Add an is_featured TINYINT(1) flag on [products]
- Add comments explaining each table's purpose

Please generate the complete SQL script I can paste into phpMyAdmin.
```

→ Review the output: check data types, foreign key references, and that sample data makes sense for your business.

Prompt 2B: Validate the Schema

```
Review this SQL script for errors and best practices:
```

```
[PASTE YOUR SQL SCRIPT HERE]

Check for:
1. Are tables created in the right order (parent tables before children)?
2. Do foreign keys reference the correct columns?
3. Are data types appropriate (DECIMAL for money, not FLOAT)?
4. Is the bcrypt hash for the admin password actually valid?
5. Does the sample data match the column definitions?
6. Are there any missing NOT NULL constraints on required fields?
```

→ Always validate AI-generated SQL before running it. This is your safety net.

Step 3: Generate the Setup Script

This is the clever part — a single PHP script that generates your entire website. Students upload one file, run it in a browser, and the whole site appears.

Prompt 3A: PHP Setup Script Generator

```
Create a PHP setup script (setup.php) that, when run in a browser,
generates a
complete website structure for my [YOUR BUSINESS NAME] web application.

The script should create these files in the current directory:

1. config.php — Database connection using PDO with placeholders for:
   - $db_host = "localhost"
   - $db_name = "YOUR_DATABASE_NAME"
   - $db_user = "YOUR_USERNAME"
   - $db_pass = "YOUR_PASSWORD"
   Include a helper function h() for htmlspecialchars() and
session_start().

2. css/style.css — Complete stylesheet with:
   - CSS variables for a [describe color palette: warm cream, rose gold,
coral accents]
   - Styles for: header, navigation, hero banner, product grid, product
cards,
     buttons, forms, data tables, messages, footer
   - Admin-specific styles (dark header, card layout, login box)
   - Responsive breakpoints for mobile
   - Use [Google Font names or describe the feel: elegant serif + clean
sans-serif]

3. includes/header.php — Shared page header with logo and navigation
4. includes/footer.php — Shared page footer with copyright and admin link
5. includes/admin_header.php — Admin header with session check (redirect
if not logged in)

6. Customer pages:
   - index.php — Homepage showing featured [products] from database
   - [products].php — Full listing with category filter buttons
```

```
   - [product]_detail.php — Single item view using JOIN (products +
categories + suppliers)
   - about.php — About page with brand story
   - newsletter.php — Newsletter signup form (INSERT with prepared
statement)

7. Admin pages:
   - admin/login.php — Login form with password_verify()
   - admin/logout.php — Destroy session and redirect
   - admin/dashboard.php — Statistics using COUNT, SUM, COALESCE
   - admin/[products].php — List all items + edit form (UPDATE with
prepared statement)
   - admin/supplier_report.php — Report using LEFT JOIN, GROUP BY, COUNT,
AVG, MIN, MAX, SUM

8. .htaccess — Disable directory browsing, block direct access to
config.php

The setup script should show a success page listing all created files when
done.
Include clear comments throughout ALL generated code explaining what each
section does.

Target audience: Beginner students deploying to shared hosting
(Hostinger).
Root folder: public_html/
```

→ This is the most complex prompt. The AI may need to generate it in parts. Ask for the setup script first, then review each generated file.

Step 4: Add Shopping Cart & E-Commerce

This step adds the ability for customers to create accounts, add items to a cart, and place orders. The cart is session-based; orders are stored in the database.

Prompt 4A: E-Commerce Database Tables

```
Add e-commerce tables to my existing [YOUR BUSINESS NAME] MySQL database.

Create these new tables:

1. customers — Registration/login with:
   - customer_id (PK), first_name, last_name, email (UNIQUE),
password_hash
   - phone, address_line1, address_line2, city, state, zip_code
   - created_at, updated_at

2. orders — Order header with:
   - order_id (PK), customer_id (FK → customers)
   - order_date, status
ENUM('pending','processing','shipped','delivered','cancelled')
   - subtotal, tax_amount, total_amount (all DECIMAL)
   - shipping_name, shipping_address, shipping_city, shipping_state,
shipping_zip
   - payment_method DEFAULT 'simulated', payment_status
ENUM('pending','paid','refunded')

3. order_items — Line items with:
   - item_id (PK), order_id (FK → orders), product_id (FK → [products])
   - quantity, price_each (snapshot at purchase time), line_total

Include 3 sample customers (password "customer123" bcrypt hashed),
4 sample orders in different statuses, and matching order_items.
```

→ The price_each snapshot is important — explain to the AI that you store the price at time of purchase because product prices can change later.

Prompt 4B: Cart & Checkout PHP Files

```
Create the PHP files for a shopping cart and checkout system for my
[YOUR BUSINESS NAME] web application.

Build these files:

1. includes/CartHelper.php — A PHP class for session-based cart:
   - add(), update(), remove(), clear(), getCount(), isEmpty()
   - getDetailedItems($pdo) — JOINs cart product IDs with the database
   - getSubtotal($pdo) — Calculates total from database prices

2. Updated includes/header.php — Add cart icon with item count badge,
   Sign In / My Account link

3. customer_register.php — Registration form with password_hash()
4. customer_login.php — Login with password_verify(), redirect support
5. customer_logout.php — Unset customer session keys only

6. add_to_cart.php — POST handler, validates product exists, redirects
(PRG pattern)
7. cart.php — Display cart, update quantities, remove items, show
subtotal/tax/total

8. checkout.php — CRITICAL: This must use a DATABASE TRANSACTION:
   - Require customer login
   - Collect shipping address (pre-fill from customer profile)
   - Simulated payment fields (with clear "no real charges" notice)
   - BEGIN TRANSACTION → INSERT order → INSERT order_items → UPDATE stock
→ COMMIT
   - ROLLBACK on any failure
   - Clear cart and redirect to confirmation

9. order_confirmation.php — Show order details using JOIN
10. my_orders.php — Customer order history with GROUP BY for item counts

Use prepared statements everywhere. Include teaching comments explaining
the transaction pattern, the PRG pattern, and the price snapshot concept.
```

→ The transaction in checkout.php is the most important teaching moment. Make sure the AI includes BEGIN, COMMIT, and ROLLBACK with clear comments.

Step 5: Add Admin Order Management & Sales Reports

Prompt 5A: Admin Orders & Sales Pages

```
Create admin pages for order management and sales reporting for my
[YOUR BUSINESS NAME] web application.

1. admin/orders.php — Order management page:
   - Display all orders with customer names (JOIN orders + customers)
   - Item count per order using a subquery: (SELECT COUNT(*) FROM
order_items...)
   - Filter buttons by status (pending, processing, shipped, delivered,
cancelled)
   - Inline status update dropdown with UPDATE query
   - Summary stats at top using CASE WHEN for conditional counting
   - Color-coded status badges

2. admin/sales_report.php — Revenue analytics page:
   - Summary stats: total orders, unique customers (COUNT DISTINCT),
     total revenue, average order value, largest order
   - Revenue by product: JOIN order_items + products + categories,
     GROUP BY product, SUM line_total
   - Revenue by category: GROUP BY category with doughnut chart
   - Top customers by spending: GROUP BY customer, SUM total_amount
   - Use Chart.js for visualizations (load from CDN)
   - Pass PHP data to JavaScript using json_encode()

Both pages require admin login (include admin_header.php).
Use the existing CSS styling. Include teaching comments explaining
each SQL technique: multi-table JOINs, GROUP BY, aggregates, CASE WHEN,
subqueries.
```

→ The sales report is the most SQL-heavy page in the project. It's a great portfolio piece for interviews.

Step 6: Add an External Data Analytics Dashboard

This step teaches students how to pull data from free government APIs (no API key required), cache it in MySQL, and visualize it — a real-world business intelligence pattern.

Prompt 6A: API Caching System

```
Create a PHP class (includes/ApiHelper.php) that fetches data from
external
APIs and caches the responses in a MySQL table.

The class needs:
- A constructor that takes a PDO connection and cache TTL (default 24
hours)
- fetchWithCache($url, $cache_key) — Check cache first, call API if
expired
- Private callApi($url) — Uses file_get_contents() with 15-second timeout
- Private getCache($key) — SELECT from api_cache WHERE expires_at > NOW()
- Private setCache($key, $data) — INSERT ON DUPLICATE KEY UPDATE
- clearExpiredCache() — DELETE WHERE expires_at < NOW()

- fetchCensusStateData($year) — Calls the U.S. Census ACS 5-Year API:
  URL: https://api.census.gov/data/{year}/acs/acs5?
get=NAME,B01003_001E,B19013_001E,B01002_001E,B01001_026E&for=state:*
  Parses the response into an array of states with population,
median_income, median_age, female_pct

- fetchBlsSeries($series_ids, $start_year, $end_year) — Calls BLS API v1:
  URL: https://api.bls.gov/publicAPI/v1/timeseries/data/{seriesID}?
startyear={}&endyear={}
  Returns chronologically sorted monthly data points

- saveCensusData($data) and saveBlsData($data) — Save to structured MySQL
tables

Also create the SQL for 3 new tables: api_cache, census_demographics,
bls_employment.

Include detailed teaching comments explaining the caching pattern.
```

→ The Census API allows 500 queries/day without a key. BLS v1 has no registration requirement.

Prompt 6B: Analytics Dashboard & Fetch Page

```
Create two admin PHP pages for my [YOUR BUSINESS NAME] analytics
dashboard:

1. admin/fetch_data.php — Data control panel:
   - Button to fetch Census data (with year selector)
   - Button to fetch BLS data (unemployment, CPI, [industry-relevant]
prices)
   - Button to clear expired cache
   - Data Status table showing record counts and last fetch time
   - "How It Works" teaching section explaining the data flow

2. admin/analytics.php — Dashboard (reads from LOCAL MySQL, not API):
   - Quick stats row using SUM, AVG, COUNT aggregate queries
   - Top 15 [states/markets] by population (bar chart)
   - Top 15 by median income (bar chart)
   - BLS monthly trend line charts
   - Target Market Analysis table with a weighted scoring formula:
     (income/max_income)*40 + (population/max_pop)*35 + (female_pct/55)*25
     This uses subqueries for normalization
   - SQL techniques box explaining ORDER BY+LIMIT, aggregates, calculated
     columns, and subqueries

Replace [industry-relevant] with a BLS series related to your business:
  - Skincare/beauty: CUSR0000SAM2 (Personal Care Products)
  - Food/restaurant: CUUR0000SAF1 (Food at Home)
  - Retail: CUUR0000SAA (Apparel)

Use Chart.js. Pass data from PHP to JS using json_encode().
```

→ Adapt the BLS series IDs to match your business. The target market scoring formula can be customized with different weights.

Step 7: Design the Brand & Create Specialty Pages

Now that the functionality is built, make it look professional. These prompts help with visual design and marketing-style pages.

Prompt 7A: Generate Brand Images

```
I'm building a fictional brand called [YOUR BUSINESS NAME], a [business
type].

Generate a homepage hero banner image with:
- A professional-looking person related to the business (e.g., holding a
product)
- The brand name "[YOUR BUSINESS NAME]" in an elegant script font
- A tagline: "[YOUR TAGLINE]"
- A "Shop Now" button in [color]
- Product bottles/items visible in the composition
- Warm, luxurious aesthetic with [describe colors: marble, gold, cream]
- Landscape orientation, suitable for a full-width website banner

Also generate a product showcase image with:
- 2-3 hero products side by side
- Clean background (marble counter, natural elements)
- Rose gold / [metallic] accents
- Brand name visible on product packaging
```

→ Use this prompt with image generation tools like Midjourney, DALL-E, or Adobe Firefly. Save as PNG and upload to your images/ folder.

Prompt 7B: Best Sellers Spotlight Page

```
Create a "Best Sellers" spotlight page (bestsellers.php) for my
[YOUR BUSINESS NAME] web application.

The page should include:
- A hero banner section with gradient background and elegant heading
- A showcase image container (displays images/products.png if it exists)
- Two side-by-side product spotlight cards that pull data from the
database
```

```
  (fetch product IDs [X] and [Y] using a JOIN across products + categories
+ suppliers)
- Each card shows: product name, category, star rating, review count,
  description from DB, ingredient/feature tags as rounded pills,
  price from DB (so admin changes reflect immediately), Add to Cart button
- A "Your [Brand] Routine" section with 4 numbered steps and arrow
connectors
- Customer review cards (4 testimonials with star ratings)
- A gradient CTA banner linking to the full product catalog

Match the existing site CSS. Use the same color variables and font
families.
Include database-driven elements so prices and stock update automatically.
```

→ Customize the routine steps and review quotes for your specific business.

Step 8: Customize the CSS & Visual Design

Prompt 8A: Complete CSS Redesign

```
Redesign the CSS stylesheet for my [YOUR BUSINESS NAME] web application to
match this brand aesthetic:

Color palette:
- Primary: [hex code] — [describe: warm copper / navy blue / forest green]
- Accent: [hex code] — [describe: coral-orange / gold / sky blue]
- Background: [hex code] — [describe: warm cream / cool gray / white]
- Text: [hex code] — [describe: deep brown / charcoal / near-black]

Typography:
- Headings: [Google Font name, e.g., Playfair Display] — [style: italic
serif]
- Body: [Google Font name, e.g., Lato] — [style: clean sans-serif]

Design elements:
- Border radius: [10px for soft / 4px for sharp / 20px for playful]
- Shadows: [subtle / dramatic / none]
- Product cards: [hover lift effect / border glow / scale up]
- Buttons: [rounded / pill-shaped / square] with [uppercase text / normal
case]
- Navigation: [sticky header / transparent over hero / solid colored]

Use CSS custom properties (:root variables) for all colors and fonts.
Include styles for: site header, navigation, hero banner, value
proposition strip,
product grid, product cards with featured badges, forms, data tables,
messages (success/error), admin header, admin cards, login box, footer.
Add responsive breakpoints at 768px and 480px.
```

→ Be specific about colors and fonts. The more detail you give, the closer the result matches your vision.

Step 9: Create Setup Documentation

Prompt 9A: Student Setup Guide

```
Create a step-by-step setup guide document for my [YOUR BUSINESS NAME]
web application starter kit.

The guide should walk a complete beginner through:

Step 1: Create the Database
- Open phpMyAdmin on their hosting account
- Create a new database and user
- Import the SQL file
- Verify tables were created (with a checkpoint)

Step 2: Upload and Run the Setup Script
- Upload setup.php to public_html/
- Visit it in a browser
- See the generated file structure (show the tree)

Step 3: Configure the Database Connection
- Edit config.php with their credentials
- Show a concrete example with sample values

Step 4: Explore the Site
- Table of URLs and what to notice on each page
- Admin login credentials
- A "Try This" prompt to edit a product and see it update

Step 5: Clean Up
- Delete setup.php (with security explanation)

Include: checkpoints after each step (green boxes),
warning callouts for common mistakes (red boxes),
a Quick Reference table mapping concepts to files,
and a Troubleshooting section covering the 6 most common errors.

Tone: Friendly, encouraging, beginner-focused.
```

→ This guide saves you dozens of support emails. Invest the time to make it

> *thorough.*

Step 10: Deploy to Hostinger & Test

The final step is getting everything live on your hosting account and testing every feature.

Prompt 10A: Deployment Checklist

```
Create a deployment and testing checklist for my [YOUR BUSINESS NAME]
web application being deployed to Hostinger shared hosting.

The checklist should cover:

PRE-DEPLOYMENT:
- [ ] Database imported successfully in phpMyAdmin
- [ ] All tables visible with correct row counts
- [ ] config.php updated with real credentials
- [ ] setup.php deleted from server

PAGE-BY-PAGE TESTING:
- [ ] Homepage loads, featured products display with correct prices
- [ ] Product listing shows all items, category filter works
- [ ] Product detail page shows JOINed data (category + supplier name)
- [ ] Newsletter signup: submit a test email, verify it appears in DB
- [ ] Admin login with default credentials
- [ ] Admin dashboard shows correct statistics
- [ ] Admin product edit: change a price, verify it updates on the front
end
- [ ] Admin supplier report displays all suppliers with aggregates

E-COMMERCE TESTING (if applicable):
- [ ] Customer registration creates account (check DB)
- [ ] Customer login works with new account
- [ ] Add to Cart from product listing and detail pages
- [ ] Cart page: update quantity, remove item, clear cart
- [ ] Checkout: fill shipping, fill simulated payment, place order
- [ ] Order confirmation shows correct items and total
- [ ] My Orders page lists the new order
- [ ] Admin Orders page shows the new order with correct status
- [ ] Admin can update order status
```

```
- [ ] Sales Report reflects the new order in charts and tables
- [ ] Product stock decreased after order

ANALYTICS TESTING (if applicable):
- [ ] Fetch Census data — success message and records in DB
- [ ] Fetch BLS data — success message and records in DB
- [ ] Analytics dashboard shows charts and target market table
- [ ] Second fetch is faster (cache hit)

Format as a clean, printable checklist.
```

→ Print this out and check off each item as you test. A systematic approach catches bugs before your instructor does.

Bonus: Portfolio & Interview Prompts

Once your project is deployed, use these prompts to prepare for job interviews where you'll present your work.

Prompt 11A: Elevator Pitch Generator

```
Help me write a 60-second elevator pitch for a job interview about the
web-database application I built.

The project: [YOUR BUSINESS NAME] — a [describe] web application built
with PHP, MySQL,
HTML/CSS, JavaScript, and Chart.js, deployed on shared hosting.

Key features:
- [list your top 5 features]

Technical highlights:
- Normalized relational database with [X] tables
- PDO prepared statements for SQL injection prevention
- Session-based authentication for both customers and admin
- Database transactions (ACID) for order processing
- External API integration with MySQL caching
- Interactive data visualizations with Chart.js
- Responsive design with Google Fonts

I want the pitch to sound confident but natural — not rehearsed.
Include 2-3 sentences I can use if they ask "What was the hardest part?"
```

→ Practice this out loud. In an interview, confidence with your own project is your biggest advantage.

Prompt 11B: Technical Interview Q&A Prep

```
Generate 10 technical interview questions someone might ask me about the
web-database application I built, along with strong answers.

Cover these areas:
1. Database design (normalization, foreign keys, why you chose certain
data types)
```

```
2. SQL queries (explain the JOIN in your supplier report, the GROUP BY in
sales)
3. Security (prepared statements, password hashing, session management)
4. Architecture (why PHP + MySQL, how the caching pattern works, MVC-like
structure)
5. Trade-offs (what would you do differently, how would this scale)

My project used: PHP 7.4+, MySQL, PDO, sessions, bcrypt, Chart.js,
World Bank/Census/BLS APIs, and was deployed on Hostinger.

Format each as: Question → Strong Answer → Bonus point to mention.
```

→ Use these to rehearse. The best interview answers connect technical decisions to business outcomes.

You Now Have Everything You Need

These 13 prompt templates cover the entire journey from a blank page to a deployed, portfolio-ready web-database application. Customize them for your business, iterate with the AI, and always validate the output. The AI writes the code — you make the decisions. That’s what makes you the engineer.