# Building a PL/SQL-Based AI Chatbot for Database Support

This guide will take you step by step through the development of an AI-driven chatbot using **PL/SQL**. The chatbot will respond to **database-related queries**, analyze **SQL execution plans**, provide **health reports**, and **log interactions** for future learning.

**Step 1: Define the File Structure**

To maintain structure and modularity, we create the following directory structure:

/plsql\_chatbot

│── /sql\_scripts

│    ├── 01\_create\_chatbot\_tables.sql

│    ├── 02\_create\_chatbot\_logs\_table.sql

│    ├── 03\_create\_oracle\_text\_index.sql

│    ├── 04\_create\_health\_metrics\_view.sql

│── /plsql\_procedures

│    ├── chatbot\_response.prc

│    ├── analyze\_sql\_execution.prc

│    ├── database\_health\_report.prc

│    ├── chatbot\_logger.prc

│── /packages

│    ├── chatbot\_pkg.pks

│    ├── chatbot\_pkg.pkb

│── /tests

│    ├── test\_chatbot.sql

**Step 2: Create Database Tables**

**2.1 Chatbot Training Data Table**

The chatbot learns from past interactions by storing questions and answers.

01\_create\_chatbot\_tables.sql

CREATE TABLE chatbot\_knowledge\_base (

    id           NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

    question     VARCHAR2(500) UNIQUE,

    answer       CLOB

);

**2.2 Chatbot Logs Table**

This table logs user interactions for learning and debugging.

02\_create\_chatbot\_logs\_table.sql

CREATE TABLE chatbot\_logs (

    log\_id        NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

    user\_query    VARCHAR2(500),

    bot\_response  CLOB,

    log\_time      TIMESTAMP DEFAULT SYSTIMESTAMP

);

**2.3 Oracle Text Index for NLP Query Processing**

We use **Oracle Text** to improve chatbot responses.

03\_create\_oracle\_text\_index.sql

BEGIN

    EXECUTE IMMEDIATE 'CREATE INDEX chatbot\_idx ON chatbot\_knowledge\_base (question)

    INDEXTYPE IS CTXSYS.CONTEXT';

END;

/

**Step 3: Define Health Monitoring View**

The chatbot should provide **database health reports**.

04\_create\_health\_metrics\_view.sql

CREATE VIEW db\_health\_metrics AS

SELECT

    (SELECT COUNT(\*) FROM v$session) AS active\_sessions,

    (SELECT ROUND(SUM(bytes)/1024/1024, 2) FROM v$sgastat WHERE pool='shared pool') AS shared\_pool\_mb,

    (SELECT ROUND((SUM(used\_bytes)/SUM(allocated\_bytes))\*100, 2) FROM v$sgainfo WHERE name LIKE 'SGA%') AS memory\_usage\_pct,

    (SELECT ROUND((SELECT value FROM v$sysmetric WHERE metric\_name = 'CPU Usage Per Sec'), 2)) AS cpu\_usage\_pct

FROM dual;

**Step 4: Create PL/SQL Procedures**

**4.1 Chatbot Response Handler**

This procedure processes user input and returns an appropriate response.

chatbot\_response.prc

CREATE OR REPLACE PROCEDURE chatbot\_response(p\_user\_query IN VARCHAR2, p\_response OUT CLOB) IS

    v\_best\_match VARCHAR2(500);

    v\_answer CLOB;

BEGIN

    -- Try to find an exact match in the knowledge base

    SELECT answer INTO v\_answer

    FROM chatbot\_knowledge\_base

    WHERE LOWER(question) = LOWER(p\_user\_query);

    p\_response := v\_answer;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        -- Use Oracle Text to find the closest matching question

        SELECT question INTO v\_best\_match

        FROM chatbot\_knowledge\_base

        WHERE CONTAINS(question, p\_user\_query) > 0

        FETCH FIRST 1 ROW ONLY;

        SELECT answer INTO v\_answer FROM chatbot\_knowledge\_base WHERE question = v\_best\_match;

        p\_response := v\_answer;

    WHEN OTHERS THEN

        p\_response := 'I am unable to process your query at the moment. Please try again later.';

END chatbot\_response;

/

**4.2 SQL Execution Plan Analyzer**

This procedure **analyzes an SQL query's execution plan** and provides optimization suggestions.

analyze\_sql\_execution.prc

CREATE OR REPLACE PROCEDURE analyze\_sql\_execution(p\_sql\_query IN VARCHAR2, p\_suggestions OUT CLOB) IS

    v\_plan\_table SYS.DBMS\_XPLAN\_TYPE\_TABLE;

BEGIN

    -- Generate an execution plan

    EXECUTE IMMEDIATE 'EXPLAIN PLAN FOR ' || p\_sql\_query;

    -- Fetch the plan

    SELECT \* BULK COLLECT INTO v\_plan\_table FROM TABLE(DBMS\_XPLAN.DISPLAY);

    -- Analyze for optimization suggestions

    p\_suggestions := 'Execution Plan: ' || CHR(10) || LISTAGG(v\_plan\_table.plan\_table\_output, CHR(10)) WITHIN GROUP (ORDER BY NULL);

    IF p\_suggestions LIKE '%FULL TABLE SCAN%' THEN

        p\_suggestions := p\_suggestions || CHR(10) || 'Optimization Tip: Consider adding indexes to avoid full table scans.';

    END IF;

EXCEPTION

    WHEN OTHERS THEN

        p\_suggestions := 'Error analyzing execution plan: ' || SQLERRM;

END analyze\_sql\_execution;

/

**4.3 Database Health Report**

This procedure generates a **database health summary**.

database\_health\_report.prc

CREATE OR REPLACE PROCEDURE database\_health\_report(p\_report OUT CLOB) IS

    v\_sessions NUMBER;

    v\_memory NUMBER;

    v\_cpu\_usage NUMBER;

BEGIN

    -- Fetch health metrics

    SELECT active\_sessions, shared\_pool\_mb, memory\_usage\_pct, cpu\_usage\_pct

    INTO v\_sessions, v\_memory, v\_cpu\_usage

    FROM db\_health\_metrics;

    -- Construct the report

    p\_report := 'Database Health Report:' || CHR(10) ||

                '- Active Sessions: ' || v\_sessions || CHR(10) ||

                '- Shared Pool Memory: ' || v\_memory || ' MB' || CHR(10) ||

                '- Memory Usage: ' || v\_cpu\_usage || '%' || CHR(10);

EXCEPTION

    WHEN OTHERS THEN

        p\_report := 'Error generating health report: ' || SQLERRM;

END database\_health\_report;

/

**4.4 Chatbot Logger**

This procedure **logs user queries and responses**.

chatbot\_logger.prc

CREATE OR REPLACE PROCEDURE chatbot\_logger(p\_user\_query IN VARCHAR2, p\_bot\_response IN CLOB) IS

BEGIN

    INSERT INTO chatbot\_logs (user\_query, bot\_response)

    VALUES (p\_user\_query, p\_bot\_response);

    COMMIT;

END chatbot\_logger;

/

**Step 5: Create Chatbot Package**

To encapsulate the chatbot logic, we create a **PL/SQL package**.

chatbot\_pkg.pks

CREATE OR REPLACE PACKAGE chatbot\_pkg AS

    PROCEDURE process\_chatbot\_query(p\_user\_query IN VARCHAR2, p\_final\_response OUT CLOB);

END chatbot\_pkg;

/

chatbot\_pkg.pkb

CREATE OR REPLACE PACKAGE BODY chatbot\_pkg AS

    PROCEDURE process\_chatbot\_query(p\_user\_query IN VARCHAR2, p\_final\_response OUT CLOB) IS

        v\_response CLOB;

    BEGIN

        chatbot\_response(p\_user\_query, v\_response);

        -- Log the interaction

        chatbot\_logger(p\_user\_query, v\_response);

        p\_final\_response := v\_response;

    END process\_chatbot\_query;

END chatbot\_pkg;

/

**Step 6: Test the Chatbot**

To ensure functionality, we run tests.

test\_chatbot.sql

SET SERVEROUTPUT ON;

DECLARE

    v\_response CLOB;

BEGIN

    chatbot\_pkg.process\_chatbot\_query('What is an index in Oracle?', v\_response);

    DBMS\_OUTPUT.PUT\_LINE('Chatbot Response: ' || v\_response);

END;

/

**Conclusion**

**Processes User Queries** (Natural Language Processing using Oracle Text).  
**Analyzes SQL Execution Plans** (Provides optimization tips).  
**Generates Health Reports** (Tracks active sessions, memory usage, CPU load).  
**Logs Interactions** (For learning and debugging).