

Project Proposal

Project Title: Developing an NLP-Based Model for Generating Intelligent Administrative and Executive Reports within an Oracle APEX Environment: An Applied Study

Student: Eman Alhabashi

Department: Department of Artificial Intelligence

University Name: MidOcean University

Table of Contents

Project Proposal	1
Project Overview	4
Project Goal	4
Main objectives:	4
Approach	5
1. Prepare the Data	5
2. Adjust the Data	5
3. Choosing the Model and Building the System.....	5
4. Testing	6
Project Steps and Timeline.....	6
References.....	8

Developing an NLP-Based Model for Generating Intelligent Administrative and Executive Reports within an Oracle APEX Environment: An Applied Study

Project Overview

Most organizations and governmental institutions are highly dependent on fixed dashboards and tables to derive decisions or insights. The same goes with Oracle APEX — a widely used low-code platform for data-driven applications — systems and applications. Every new question asked by the decision-makers must go through three steps: the executive asks the question, current reports display raw data, and the developer designs and deploys a new report or chart.

This paper proposes a layer in APEX that addresses the above-mentioned challenges. The decision-makers ask the question in their preferred language(Arabic or English) and receive a correct validate answers. This NLP layer will translate the asked questions into SQL (Natural Language to SQL, NL2SQL) and then transfer the SQL result (or report) into a narrative summary and insight (data-to-text generation), allowing decision-makers to understand the answer to their questions.

The importance of this work is both practical and academic. Practically, it helps decision-makers, reduces dependence on developers, and offers insights that non-experts might otherwise miss; the bilingual focus is directly relevant to the Arab speaking countries, where executive reporting is frequently bilingual and where Arabic remains under-resourced for NL2SQL. Academically, this is an applied study: rather than evaluating a model in isolation, it integrates NLP into a real, deployable APEX application, contributing an architecture for intelligent reporting in low-code environments.

Project Goal

We aim in this project to design and build an NLP-based model that generates intelligent reports from written questions and raw data using Oracle APEX environment

Main objectives:

1. **Develop** an **NL2SQL** module that translates questions into SQL in any Oracle database.
2. **Develop** a **data-to-text (NLG)** module to generate written insights from the previously created queries
3. **Build** an **Oracle APEX** application as a prototype to display the whole process: Question => SQL => result => written report.
4. **Evaluate** with target end-users

Approach

This project has four main parts: preparing the data, adjusting the data for the model, choosing and building the model, and finally testing.

1. Prepare the Data

To achieve the purpose of this project, we need two distinct datasets. The first is the data the application will actually report on. For this, we will set up a realistic database in Oracle—using Oracle's own sample schemas (HR and Sales History). We will also need to write clear descriptions for each table and column inside the database(meta-data), which the model relies on.

The second set of data is for training and testing the model. For translating questions into SQL, there are well-known public datasets we can use — Spider, BIRD, and Spider 2.0 (which will be very helpful since it is built from messy, real company databases rather than clean testing scenarios). For transferring data into written summaries, we will use standard table-to-text datasets such as ToTTo, DART, and WebNLG.

Finally, we will build our own small test set with real end-users (decision-makers). Each item will pair an actual executive question, written in both Arabic and English, with the correct SQL and a well-written answer, which will allow us to measure the efficiency of this project when facing questions in practice, in both languages.

2. Adjust the Data

Before the model can answer a question, it needs to know what the database looks like. So, the first step is to pull the table names, column names, descriptions, and relationships out of Oracle and turn them into a form the model can read. I will pair this with a few worked examples, so the model has something to imitate.

Arabic needs extra care here. Arabic text comes in slightly different written forms for the same word, so we will clean and standardize it first; otherwise, the model treats "the same" question as two different ones.

Finally, once a query returns results, we will write those results into a clear format — with the right units and comparisons — so that when the system writes the summary, it is working from clean numbers and is less likely to make something up.

3. Choosing the Model and Building the System

For turning questions into SQL, we will use a large language model guided by prompting, the worked examples, and the relevant history we just described. Oracle already ships its own natural-language-to-SQL feature (Select AI, plus the APEX AI Assistant), so they will be used as a starting point and comparison tool.

No query must run blindly: the system generates the SQL, shows it (with a plain explanation of what it does), and a person confirms it before anything executes. We could also test whether a smaller open model like Llama 3 or Qwen can do the job, since that would be cheaper and could run inside the organization rather than relying on an outside service.

For writing the report itself, the model takes the question and the cleaned-up results and produces a short, plain-language summary. The main risk here is the model stating things the data does not support, so we will constrain it to only describe what is actually in the numbers and have its reason step by step before writing.

The final scenario will be the user asks a question in APEX, the question goes to the language model, the model proposes SQL, the system checks the SQL is safe and runs it, the results come back, and finally the model writes them up as a short report shown on the screen. Throughout, the system only ever sees the data the user is already allowed to see.

4. Testing

The testing will depend on 2 main conditions.

First, is the generated SQL correct? We will measure how often the generated query returns the correct answer, both on the public datasets and on my own Arabic/English test set, and how fast it responds.

Second, are the written summaries accurate and readable? The main concern is whether the text stays true to the data, so we will check it via automatic measures and with human reviewers. We will also check the clarity and usefulness of the summaries.

Project Steps and Timeline

Estimated total duration: approximately 6-7 months. According to the following table:

Phase	Key Activities	Timeframe
Proposal & literature review	Finalize the proposal, review NL2SQL and data-to-text literature, and confirm scope and datasets.	Weeks 1–4
Data preparation	Set up the Oracle database with metadata; collect datasets, begin the bilingual test set.	Weeks 4–7
Data adjustment	Extract the schema into a model-readable form, normalize Arabic text, and format query results.	Weeks 7–9

Phase	Key Activities	Timeframe
NL2SQL module & baseline	Build the prompt-guided NL2SQL module and compare it against Oracle Select AI and the APEX AI Assistant.	Weeks 9–13
Data-to-text (NLG) module	Build the summary generator, constrained to the data and reasoning step by step before writing.	Weeks 13–18
APEX integration	Assemble scenario (Question → SQL → safety check → run → report) with human confirmation.	Weeks 18–20
Testing & evaluation	Measure SQL accuracy; check summary with human reviewers.	Weeks 20–23
Comparison & analysis	Compare Oracle’s built-in AI features with the proposed approach, especially in Arabic.	Weeks 23–25
Final write-up & defense	Document the thesis and prepare the defense.	Weeks 25–26

Expected Results

By the end of the project, the main expectation is to have a working tool inside Oracle APEX where the manager should be able to type a question in Arabic or English, and the system will generate the correct query, run it, and display both the numbers and a short written summary explaining their meaning.

We also expect to have a real measuring on the efficiency of query generation on both the public test sets and my own Arabic/English set, and how accurate and trustworthy the written summaries are.

The project should also produce some useful comparisons. We would like to see the difference between Oracle AI and our approach, especially in Arabic, since Arabic has far less support.

Finally, the project is meant to leave something behind that others can build on: a clear blueprint for adding this kind of natural-language reporting to other APEX applications, a bilingual Arabic–English test set for administrative questions that did not exist before, and a set of practical lessons on how to keep the system accurate and secure.

Resources Needed

1. Oracle platform: Oracle Database 23ai or Autonomous Database (OCI Always Free tier is sufficient for development), Oracle APEX (apex.oracle.com or a local instance).
2. LLM access: an API to a commercial model (e.g., OpenAI, Cohere, or OCI Generative AI) and, optionally, open models (Llama 3, Qwen).
3. Development tools: VS Code, SQL Developer / SQLcl, Python (transformers, datasets, evaluation libraries), Git.
4. Datasets: Spider, BIRD, Spider 2.0; ToTTo, DART, WebNLG; Oracle sample schemas; plus the custom bilingual set.

References

- Yu, T., Zhang, R., Yang, K., et al. (2018). Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. EMNLP 2018.
- Li, J., Hui, B., Qu, G., et al. (2023). Can LLM Already Serve as a Database Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs (BIRD). NeurIPS 2023.
- Lei, F., Chen, J., Ye, Y., et al. (2025). Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. ICLR 2025.
- Hong, Z., Yuan, Z., Zhang, Q., et al. (2024). Next-Generation Database Interfaces: A Survey of LLM-Based Text-to-SQL. arXiv:2406.08426.
- Shi, L., Tang, Z., Zhang, N., et al. (2024). A Survey on Employing Large Language Models for Text-to-SQL Tasks. arXiv:2407.15186.
- Gao, D., Wang, H., Li, Y., et al. (2024). Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation (DAIL-SQL). Proceedings of the VLDB Endowment.
- Lin, Y., Ruan, T., Liu, J., & Wang, H. (2024). A Survey on Neural Data-to-Text Generation. IEEE Transactions on Knowledge and Data Engineering, 36(4), 1431–1449.
- Li, W., Wu, W., Chen, M., et al. (2022). Faithfulness in Natural Language Generation: A Systematic Survey of Analysis, Evaluation and Optimization Methods. arXiv:2203.05227.
- Zhao, Y., et al. (2023). Investigating Table-to-Text Generation Capabilities of LLMs in Real-World Information Seeking Scenarios. arXiv:2305.14987.

- Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS 2020.
- Oracle. Configuring Generative AI in Oracle APEX using Select AI and DBMS_CLOUD_AI documentation. Oracle APEX Blog / Oracle Documentation.
- Oracle. Announcing Oracle APEX 24.2 General Availability — Generative Development (GenDev), AI Configurations, and RAG Sources. Oracle APEX Blog, 2024–2025.
- Oracle. Oracle APEX AI Platform. <https://apex.oracle.com/en/platform/ai/>