

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

Group Members:

Eman Alhabashi

Merwa Idris

Alaa Humaid

Department: Department of Artificial Intelligence

University Name: MidOcean University

Table of Contents

Chapter 1	4
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
Expected Results	7
Resources Needed.....	8
Chapter 2: Literature Review	9
2.1 Introduction.....	9
2.2 Asking questions in natural language: NL2SQL(5) (2).....	9
2.3 Turning data into written reports: text generation (9).....	11
2.4 Bringing AI into Oracle APEX and low-code platforms (4)	13
2.5 Synthesis and research gap	15
Appendix: Synthesis Matrix.....	16
References	18

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

Chapter 1

1.1 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.

1.2 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

1.3 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.

1.4 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

1.5 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.

1.6 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.

Chapter 2: Literature Review

2.1 Introduction

Most organizations already hold the data they need to run themselves. It lives in relational databases behind their day-to-day systems: finance, human resources, operations, case records. The difficulty is rarely a shortage of data. It is getting an answer out of that data and into a shape a manager can use without waiting for a developer to build yet another report. Two capabilities from natural language processing speak to this. One lets a person ask a question in plain language and have the system write the database query for them. The other takes the rows that come back and turns them into a short written account instead of leaving the reader to interpret a table. This thesis works at the join between those two, and adds a condition most studies leave out: keeping the whole thing inside a low-code platform, Oracle APEX, where the data and the application already sit together.

This chapter reviews the work that bears on that goal across three areas. The first is converting natural language into SQL, which has the most active research record. The second is generating text from data, where the recurring worry is whether the words stay true to the numbers. The third, and youngest, is the effort to put AI to work inside low-code tools and APEX in particular. I have kept the focus on work from 2020 onward, since the field moved sharply once large language models arrived and older results no longer describe what these systems can do.

2.2 Asking questions in natural language: NL2SQL(5) (2)

The dominant way to frame this problem is as translation: take a question and the structure of the database it refers to, and produce a SQL statement that runs and returns the right answer. A fluent model will gladly produce SQL that does not parse, so some of the early progress came from controlling the output. Scholak, Schucher, and Bahdanau (2021), in PICARD, constrained the model's decoding so that at each step it could only add tokens that keep the query valid, which cut down on broken SQL without retraining the model underneath.

Large language models then shifted the methods. Instead of training a dedicated parser, researchers found they could hand a general model the schema and a few examples and get usable SQL back, provided they structured the prompt well. Pourreza and Rafiei (2023), in DIN-SQL, showed that splitting the task into smaller steps and letting the model review and repair its own draft raised accuracy well above plain prompting. Gao et al. (2024), behind DAIL-SQL, stepped back and asked which prompt choices actually matter, comparing how the question and schema are represented and how in-context examples are picked and ordered. Their contribution reads less like a new model and more like a careful account of what works, which is what a builder needs.

The benchmarks, meanwhile, became more honest about real conditions. Li et al. (2023), in BIRD, assembled large databases with messy values and a need for outside knowledge, and they measured efficiency rather than only whether the answer was right. The gap they reported between the best models and human experts was wide. Lei et al. (2025), in Spider 2.0, pushed further toward the sprawling schemas and multi-step analysis an analyst meets at work, and current models clear only a modest fraction of those tasks. Two surveys, Hong et al. (2024) and Shi et al. (2024), organize this fast-moving area, usually along one line: approaches that steer a general model through prompting, and approaches that specialize one through fine-tuning, each trading off cost, control, and data.

Read together, a few things stay unresolved for anyone building an administrative reporting tool. The benchmarks are almost all English and drawn from generic domains, so a high score says little about a specific organization's tables and vocabulary. Performance on messy, enterprise-scale data is still weak, which is the BIRD and Spider 2.0 result. The safety of a generated query, whether it only reads, whether it respects a user's permissions, and how a person checks it before it runs, is treated as an implementation detail in most academic work even though it decides whether the thing can be deployed at all. And the bilingual case, Arabic next to English, barely appears.

2.3 Turning data into written reports: text generation (9)

The second capability is the reverse direction: start from structured data and produce readable text. It helps to set the scene with the broader shift in text generation that Sharma et al. (2025) describe. Their overview traces the move from handcrafted rules to statistical methods to neural models and then transformers, and it illustrates the modern end of that line with a small experiment in which Google's Gemini turns a user's background into career insights and draft documents. The paper is introductory and runs no real evaluation, but it captures the point that matters here: contextual, transformer-based generation has largely replaced the older template and feature-engineering approaches.

Within data-to-text specifically, the work has circled one problem: keeping the text faithful to the source. Parikh et al. (2020), in ToTTo, built a controlled task in which particular table cells are highlighted and the target sentence is produced by trimming a real Wikipedia sentence down to only what the data supports, which gives a cleaner test of whether a model stays grounded. Nan et al. (2021), in DART, framed generation from sets of structured records into sentences. Two surveys then consolidate the concern itself. Li et al. (2022) review how faithfulness is analyzed, measured, and improved, and Ji et al. (2023) give a broad map of hallucination across generation tasks, data-to-text among them, with the mitigation strategies tried so far.

Large language models reopened the question of how best to chase faithfulness. Zhao et al. (2023) studied table-to-text in realistic information-seeking settings and found that LLMs can write statements more faithful than earlier fine-tuned systems, and further that an LLM can act as a reasonable judge of faithfulness and give step-by-step feedback on another model's draft. Lin et al. (2024) survey the neural data-to-text field and give it a clear shape, separating the planning of what to say from the wording of how to say it. On the applied side, Roychowdhury et al. (2024), in ERATTA, paired retrieval over enterprise tables with explicit checks for hallucinated answers, which is close to what a trustworthy reporting tool needs.

One paper here sits squarely in the domain this thesis cares about. Oyewole et al. (2024) review the use of NLP to automate financial reporting, covering sentiment and thematic

analysis, summarization, and the regulatory side, and they report gains in efficiency and accuracy alongside real worries about consistency. They cite, for instance, a query-based summarization system reaching roughly a ROUGE-1 score of 46%, and tools built specifically to test whether text-based financial forecasts stay logically consistent. It is a secondary review rather than a built system, and its prose leans heavily on the promotional register, but it earns its place for one reason: in finance and administration the cost of an unfaithful sentence is high, which is exactly why faithfulness cannot be an afterthought.

Yagamurthy, Azmeera, and Khanna (2023) take this a step closer to the administrative domain. Their study targets the automated transformation of structured organizational data — financial records, business-intelligence figures, healthcare KPIs, and customer feedback — into human-readable reports and narratives. The pipeline they describe runs through data cleaning, standardization, and data-to-text mapping before generation, and the challenges they name — domain-specific language, contextual understanding, data quality, and ethical handling — are exactly the obstacles an executive-reporting tool faces. It is not an APEX study, and it does not address bilingual output, but it is the clearest existing precedent for automated narrative report generation from the kind of data this thesis works with.

One paper here sits squarely in the domain this thesis cares about. Oyewole et al. (2024) review the use of NLP to automate financial reporting, covering sentiment and thematic analysis, summarization, and the regulatory side, and they report gains in efficiency and accuracy alongside real worries about consistency. They cite, for instance, a query-based summarization system reaching roughly a ROUGE-1 score of 46%, and tools built specifically to test whether text-based financial forecasts stay logically consistent. It is a secondary review rather than a built system, and its prose leans heavily on the promotional register, but it earns its place for one reason: in finance and administration the cost of an unfaithful sentence is high, which is exactly why faithfulness cannot be an afterthought.

The gaps mirror the query side. Faithfulness is still the open risk, and it matters most in executive reporting, where one wrong figure in a summary can sink trust in the whole tool. Most of the work is English and built on Wikipedia tables or generic sources, with little on

administrative or financial narratives, less on bilingual generation, and little that ties the written summary back to a live query result inside a working application rather than a fixed benchmark table.

2.4 Bringing AI into Oracle APEX and low-code platforms (4)

The third area is the youngest and the thinnest. Low-code platforms let people with limited programming background build working applications through visual, declarative tools, and Oracle APEX is a prominent case, set apart by running directly on the Oracle Database and shipping with it. Oracle has recently added generative AI features to the platform, and a handful of papers have begun to study what that enables.

Yeruva (2025) gives a wide review of AI inside APEX, covering predictive analytics, sentiment analysis, recommendation engines, NLP-driven interfaces, in-database machine learning, and the pattern of calling external models through REST APIs. It is broad rather than deep, with no single system built and tested end to end, and it closes on familiar challenges around integration, explainability, and scalability. Its value is as a map of the territory.

Keshireddy (2025c) go a step further by embedding a large language model directly into the APEX development environment to provide intelligent code assistance. The system helps developers generate, understand, and optimize code, and the reported outcomes include faster development cycles, fewer coding errors, and better support for the low-code model overall. Like Yeruva's review, the focus is on building applications rather than querying or reporting on their data, but it contributes something useful: it shows that LLM assistance inside APEX is not just possible at a proof-of-concept level but measurably accelerates real development work.

Keshireddy(2025d) present an AI assistant built inside Oracle APEX using Retrieval-Augmented Generation (RAG). Rather than relying on a model's fixed training knowledge, the system retrieves relevant content from organizational documents and records at query time and uses it to produce context-aware, accurate responses. The reported gains are higher response accuracy, better context understanding, and improved user satisfaction. RAG is a method this thesis relies on directly — using Oracle 23ai's vector search to pull relevant

history and schema documentation into each prompt — so this paper is direct evidence that the integration is viable inside APEX, not just in stand-alone LLM settings.

Keshireddy (2025a) is the most thorough of the APEX papers. It proposes a modular NLP architecture for APEX built around intent recognition and context-aware responses, with lightweight pipelines for intent detection, entity recognition, and adaptive dialogue. It implements four use cases: triggering predefined workflows from natural language, retrieving data on request, filling forms intelligently, and answering user queries through a conversational agent. And it evaluates with metrics a committee will recognize as real: task completion time, interaction error rate, intent-recognition precision, latency, and throughput. The focus is on interaction and usability rather than report generation, but as a template for how to structure and measure an APEX-plus-NLP system, it is the strongest of the set.

Keshireddy (2025b), in a separate study published in the Journal of Internet Services and Information Security (JISIS), shifts from interaction to data integrity. The paper proposes an adaptive AI framework inside Oracle APEX that automates data transformation and validation using semantic, statistical, and constraint-based techniques rather than the rigid PL/SQL rules most APEX systems rely on. It was tested against 980,000 records across HRMS, CRM, and Accounting modules, reporting transformation accuracy above 92% in HRMS and CRM and 94.6% in Accounting. The framework also incorporates encryption, access control, and audit logging, and it adapts automatically when schemas change. This connects to two objectives in this thesis: the need to handle real enterprise data reliably, and the need to ensure that the system's operations are secure and auditable. It is the most evidence-grounded of the APEX papers and the most explicitly enterprise-scale.

Ashraf Syed (2025) is the closest in subject to this thesis. It centers on the APEX AI Assistant introduced in APEX 24.1 and presents a proof-of-concept chatbot that turns natural-language prompts into executable SQL, reporting around 92% accuracy and a 50% cut in the time to produce a report, with an enterprise example to back the claim. It names data privacy, model accuracy, and skill gaps as the obstacles, and points to prompt engineering

and Oracle's cloud infrastructure as the answers. The subject is right, but the work is a single-author proof of concept, and the headline figures come without a transparent evaluation protocol, so the numbers are better read as encouraging than as established.

A broader lesson comes from outside APEX. Rao et al. (2024) built LowCoder, which pairs a visual interface with a natural-language interface for assembling machine-learning pipelines, and ran a user study comparing the two. Their finding carries well beyond their setting: the natural-language interface helped people discover operators and options they would not have found in the menus, but they still leaned on the visual layer to read and correct what the system produced. The implication for a reporting tool is direct. A natural-language front end works best when the person can still see and fix what the machine generated, which is the argument for letting a user inspect the SQL before it runs and weigh the summary against the numbers.

Taken together, the AI-in-APEX literature is small, recent, aimed more at building applications or smoothing interaction than at generating reports from data, and mostly at the proof-of-concept stage. Like the other two areas, it is almost entirely English.

2.5 Synthesis and research gap

Each area is in decent health on its own. NL2SQL has capable methods and increasingly realistic benchmarks, but it is tested mostly on clean, English, generic data, and the practical questions of validating queries on messy enterprise schemas, and of handling Arabic, are barely addressed. Data-to-text generation has reached the point where large models can write grounded summaries, yet faithfulness remains the unsolved risk, and administrative or bilingual reporting has had little attention. AI inside low-code platforms is just starting, and the work closest to this thesis, a SQL-generating chatbot in APEX, is a proof of concept without a rigorous evaluation, and it does not address narrative reporting or two languages.

The space between these is where this thesis sits. As far as this review found, no existing study brings together a validated natural-language-to-SQL front end and a faithfulness-controlled data-to-text back end, inside a low-code Oracle APEX application, serving

administrative and executive reporting in both Arabic and English. The pieces exist separately, usually tested far from a working enterprise system. Building and evaluating the whole pipeline in its real setting is the contribution this thesis aims for, and it is the kind of contribution an applied study, backed by direct APEX experience, is well placed to make.

Appendix: Synthesis Matrix

Work	Area	What it does	Language	Limitation for this thesis
Scholak et al. (2021), PICARD	NL2SQL	Constrained decoding for valid SQL	English	Validity only; not permissions or safety
Pourreza & Rafiei (2023), DIN-SQL	NL2SQL	Decomposition + self-correction (LLM)	English	No bilingual or deployment study
Gao et al. (2024), DAIL-SQL	NL2SQL	Systematic prompt-strategy evaluation	English	Benchmark focus; not an applied setting
Li et al. (2023), BIRD	NL2SQL	Realistic, dirty-database benchmark	English	Wide human-model gap remains
Lei et al. (2025), Spider 2.0	NL2SQL	Enterprise analytical workflows	English	Low solve rate; no Arabic
Hong et al. (2024)	NL2SQL	Survey	English	Confirms gaps; no system
Shi et al. (2024)	NL2SQL	Survey	English	Confirms gaps; no system
Sharma et al. (2025)	Text generation	Overview + Gemini career-text demo	English	Introductory; no evaluation
Parikh et al. (2020), ToTTo	Data-to-text	Controlled, highlighted-cell generation	English	Open-domain Wikipedia tables
Nan et al. (2021), DART	Data-to-text	Record / triple-to-text generation	English	Not administrative reporting
Li et al. (2022)	Data-to-text	Faithfulness survey	English	Frames the risk; no system

Work	Area	What it does	Language	Limitation for this thesis
Ji et al. (2023)	Data-to-text	Hallucination survey	English	Frames the risk; no system
Zhao et al. (2023)	Data-to-text	LLM table-to-text + LLM-as-evaluator	English	No bilingual or live-query link
Lin et al. (2024)	Data-to-text	Survey	English	Confirms gaps; no system
Roychowdhury et al. (2024), ERATTA	Data-to-text	RAG over enterprise tables + hallucination checks	English	Q&A focus, not narrative reports
Yagamurthy et al. (2023)	Data-to-text	NLG for automated report generation from structured data	English	No APEX integration; no bilingual coverage
Oyewole et al. (2024)	Data-to-text (finance)	Review of NLP for financial reporting	English	Secondary review; no built system
Yeruva (2025)	AI + APEX	Review of AI features in APEX	English	Broad; no evaluated system
Keshireddy (2025c)	AI + APEX	LLM code assistance for APEX development	English	Developer productivity focus, not reporting
Keshireddy (2025d)	AI + APEX	RAG-based AI assistant inside APEX	English	Q&A / conversational focus, not reporting
Keshireddy (2025a)	AI + APEX	Modular NLP architecture; four use cases; evaluated	English	Interaction focus, not reporting
Keshireddy (2025b, JISIS)	AI + APEX	Adaptive AI for data transformation & validation in APEX	English	Data processing focus; not NL querying or reporting
Ashraf Syed (2025)	AI + APEX	SQL-generating chatbot proof of concept (APEX 24.1)	English	Not rigorously evaluated; not narrative or bilingual
Rao et al. (2024), LowCoder	AI + low-code	Visual + NL interface for ML pipelines; user study	English	ML pipelines, not database reporting

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/>
- Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., & Zhou, J. (2024). Text-to-SQL empowered by large language models: A benchmark evaluation. *Proceedings of the VLDB Endowment*, 17(5), 1132–1145. <https://arxiv.org/abs/2308.05389>
- Hong, Z., Yuan, Z., Zhang, Q., Chen, H., Dong, J., Huang, F., & Huang, X. (2024). *Next-generation database interfaces: A survey of LLM-based text-to-SQL*. arXiv. <https://arxiv.org/abs/2406.08426>

- Lei, F., Chen, J., Ye, Y., Cao, R., Shin, D., Su, H., Suo, Z., Gao, H., Hu, W., Yin, P., Zhong, V., Xiong, C., Sun, R., Liu, Q., Wang, S., & Yu, T. (2025). Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. *International Conference on Learning Representations (ICLR)*. <https://arxiv.org/abs/2411.07763>
- Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., Wang, B., Qin, B., Cao, R., Geng, R., Huo, N., Zhou, X., Ma, C., Li, G., Chang, K. C. C., Huang, F., Cheng, R., & Li, Y. (2023). Can LLM already serve as a database interface? A big bench for large-scale database grounded text-to-SQLs. *Advances in Neural Information Processing Systems (NeurIPS)*, 36. <https://arxiv.org/abs/2305.03111>
- Pourreza, M., & Rafiei, D. (2023). DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. *Advances in Neural Information Processing Systems (NeurIPS)*, 36. <https://arxiv.org/abs/2304.11015>
- Scholak, T., Schucher, N., & Bahdanau, D. (2021). PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 9895–9901. <https://arxiv.org/abs/2109.05093>
- Shi, L., Tang, Z., Zhang, N., Zhang, X., & Yang, Z. (2024). A survey on employing large language models for text-to-SQL tasks. *arXiv*. <https://arxiv.org/abs/2407.15186>
- Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12), 1–38. <https://arxiv.org/abs/2202.03629>
- Li, W., Wu, W., Chen, M., Liu, J., Xiao, X., & Wu, H. (2022). *Faithfulness in natural language generation: A systematic survey of analysis, evaluation and optimization methods*. *arXiv*. <https://arxiv.org/abs/2203.05227>
- 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. <https://doi.org/10.1109/TKDE.2023.3304385>
- Nan, L., Radev, D., Zhang, R., Rau, A., Sivaprasad, A., Hsieh, C., Tang, X., Vyas, A., Verma, N., Krishna, P., Liu, Y., Irwanto, N., Pan, J., Rahman, F., Zaidi, A., Mutuma, M., Tarabar, Y., Gupta, A., Yu, T., ... Radev, D. (2021). DART: Open-domain structured data record to text generation. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 432–447. <https://arxiv.org/abs/2007.02871>
- Oyewole, A. T., Adeoye, O. B., Addy, W. A., Okoye, C. C., Ofodile, O. C., & Ugochukwu, C. E. (2024). Automating financial reporting with natural language processing: A review and case analysis. *World Journal of Advanced Research and Reviews*, 21(3), 575–589. <https://doi.org/10.30574/wjarr.2024.21.3.0688>

- Parikh, A. P., Wang, X., Gehrmann, S., Faruqui, M., Dhingra, B., Yang, D., & Das, D. (2020). ToTTo: A controlled table-to-text generation dataset. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1173–1186. <https://arxiv.org/abs/2004.14373>
- Roychowdhury, S., Krema, M., Mahammad, A., Moore, B., Mukherjee, A., & Prakashchandra, P. (2024). ERATTA: Extreme RAG for table-to-answers with large language models. *IEEE International Conference on Big Data*. <https://arxiv.org/abs/2405.03963>
- Sharma, A., Giri, S., Pandey, S., Pandey, P., & Pandey, S. K. (2025). A study on natural language processing techniques for intelligent text generation. *International Journal for Research in Applied Science & Engineering Technology*, 13(12). <https://doi.org/10.22214/ijraset.2025.76052>
- Zhao, Y., Zhang, H., Si, S., Nan, L., Tang, X., & Cohan, A. (2023). Investigating table-to-text generation capabilities of large language models in real-world information seeking scenarios. *Findings of the Association for Computational Linguistics: EMNLP 2023*, 160–175. <https://arxiv.org/abs/2305.14987>
- Keshireddy, S. R. (2025). Natural language processing integration in Oracle APEX for enhanced user interaction in ubiquitous systems. *International Journal of Emerging Trends in Computer Science and Informatics*, 1(3).
- Rao, N., Tsay, J., Kate, K., Hellendoorn, V. J., & Hirzel, M. (2024). AI for low-code for AI. *Proceedings of the 29th International Conference on Intelligent User Interfaces (IUI '24)*, 837–852. <https://doi.org/10.1145/3640543.3645203>
- Syed, A. (2025). *Leveraging generative AI in Oracle APEX for intelligent applications* [Preprint]. ResearchGate. <https://www.researchgate.net/publication/397555550>
- Yeruva, S. K. (2025). Leveraging AI features in Oracle APEX for smart application development. *World Journal of Advanced Engineering Technology and Sciences*, 16(1), 624–633. <https://doi.org/10.30574/wjaets.2025.16.1.1239>
- Yagamurthy, D. N., Azmeera, R., & Khanna, R. (2023). Natural language generation (NLG) for automated report generation. *Journal of Technology and Systems*, 5(1), 48–59. <https://doi.org/10.47941/jts.1497>
- Keshireddy, S. R. (2025). Automated data transformation and validation in Oracle APEX using adaptive AI models for secure enterprise applications. *Journal of Internet Services and Information Security (JISIS)*, 15(2), 185–208. <https://doi.org/10.58346/JISIS.2025.12.041>
- Keshireddy, S.R. (2025) Retrieval-Augmented Generation Techniques in Oracle APEX: Improving Contextual Responses in AI Assistants. Available at: <https://orcid.org/0009-0007-6482-4438> (Accessed: 27 June 2026).

