

إيمان الحبشي

AI Powered Executive Reporting An Oracle APEX Implementation.pdf

 Midocean University

Document Details

Submission ID

trn:oid::2945:406312446

Submission Date

Aug 8, 2026, 12:55 PM GMT+3

Download Date

Aug 8, 2026, 6:26 PM GMT+3

File Name

AI Powered Executive Reporting An Oracle APEX Implementation.pdf

File Size

789.6 KB

42 Pages

8,663 Words

50,460 Characters

59% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely revised by AI.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.



45 AI-generated 59%

Likely generated or likely generated and revised by AI.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely human generated as AI generated and likely AI generated as human generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.



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

Prepared By

Name: Eman Al Habashi

ID No: 261000480

Name: Merwa Idris

ID No: 221002171

Name: Alaa Humaid

ID No: 261002982

Submitted in fulfillment of the requirements for the degree Master of
Science in Artificial Intelligence

At

College of Informatics

Midocean University

KM

Supervisor Name: Yasser Abdel-Hamid

College of Informatics, Midocean University, Comoros

Email:.....

Acknowledgements

We would like to express our sincere gratitude to **Dr. Yasser** for his continuous guidance, valuable advice, and constructive feedback throughout this research. Also, we would like to thank everyone who contributed directly or indirectly to the completion of this research.

Abstract

Administrative and executive reporting in enterprise and government organizations continues to depend on static, developer-built dashboards that cannot respond to ad-hoc questions and produce no narrative explanation of the data they display. This paper presents the architecture and design of a bilingual natural language processing framework embedded within Oracle Application Express (APEX) that addresses both limitations.

The proposed system accepts queries in Arabic or English, translates them into validated, read-only SQL through a prompt-engineered large language model augmented with retrieval-augmented generation and a human-in-the-loop confirmation step, executes the query, and generates a concise, faithful narrative summary through a second language model stage.

The framework is designed for deployment on Oracle Database using the HR or Sales History schemas as the evaluation domain. Key architectural contributions include a schema-linking and RAG pipeline using Oracle Search, a SQL safety and validation layer, and a faithfulness-controlled narrative generation component that supports bilingual Arabic-English output.

This paper presents the complete system architecture and design rationale, defines a rigorous evaluation framework covering NL2SQL accuracy, narrative faithfulness, and user task efficiency, and identifies the research gap this work addresses: no existing study integrates validated NL2SQL and controlled NLG in a bilingual administrative reporting context within a low-code enterprise platform. System implementation and experimental results will be reported in subsequent work.

Contents

Acknowledgements.....	3
Abstract.....	5
List of Tables.....	7
List of Figures.....	7
List of Abbreviations.....	7
Chapter 1 Introduction.....	10
1.1 Overview.....	10
1.2 Problem Statement.....	11
1.3 Research Questions.....	12
1.4 Significance of the Study.....	12
1.5 Scope and Limitations.....	13
1.6 Approach.....	14
1.7 Thesis Organization.....	15
1.8 Expected Results.....	16
1.9 Resources Needed.....	16
Chapter 2 Background and Related Work.....	18
2.1 Background.....	18
2.2 Question Answering for NL2SQL.....	18
2.3 Turning data into written reports: text generation.....	20
2.4 Bringing AI into Oracle APEX and low-code platforms.....	22
2.5 Synthesis and research gap.....	25
2.6 A Worked Example: From Question to Report.....	26
2.6.1 The Scenario.....	28
2.6.1.1 User Input.....	29
2.6.1.2 Schema Linking and Context Retrieval.....	29
2.6.1.3 NL2SQL Generation.....	30
2.6.1.4 Human Confirmation.....	32
2.6.1.5 Execution and Result Formatting.....	32
2.6.1.6 Narrative Generation.....	33
2.6.1.7 Report Display.....	33
Appendix: Synthesis Matrix.....	34
References.....	37

List of Tables

Table 1 2.6 NL2SQL Query	29
Table 2 2.6 Execution and Result Formatting	31
Table 3 2.5 Synthesis Matrix	34

List of Figures

Figure 1 2.6 System pipeline architecture for the proposed NLP-based reporting framework	26
Figure 2 2.6 Complete cycle from user input to executive report for the worked example scenario	30

List of Abbreviations

The following abbreviations are used throughout this thesis.

Abbreviation	Full Form
ACM	Association for Computing Machinery
AI	Artificial Intelligence
API	Application Programming Interface
APEX	Application Express (Oracle)
BI	Business Intelligence
BIRD	Big Bench for Large-Scale Database Grounded Text-to-SQL
BLEU	Bilingual Evaluation Understudy
CAGR	Compound Annual Growth Rate
CRM	Customer Relationship Management
DART	Open-Domain Structured Data Record to Text (dataset)
DDL	Data Definition Language
DML	Data Manipulation Language

Abbreviation	Full Form
DOI	Digital Object Identifier
EMNLP	Conference on Empirical Methods in Natural Language Processing
ERATTA	Extreme RAG for Table-to-Answers
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
HR	Human Resources (Oracle sample schema)
HRMS	Human Resource Management System
ICLR	International Conference on Learning Representations
IEEE	Institute of Electrical and Electronics Engineers
IJRASET	International Journal for Research in Applied Science and Engineering Technology
IUI	ACM Conference on Intelligent User Interfaces
JISIS	Journal of Internet Services and Information Security
KPI	Key Performance Indicator
LLM	Large Language Model
MENA	Middle East and North Africa
ML	Machine Learning
NAACL	North American Chapter of the Association for Computational Linguistics
NeurIPS	Neural Information Processing Systems
NL2SQL	Natural Language to SQL
NLG	Natural Language Generation
NLP	Natural Language Processing
OCI	Oracle Cloud Infrastructure
ORDS	Oracle REST Data Services
PARENT	Precision and Recall of Entities for Notation Translation (metric)
PICARD	Parsing Incrementally for Constrained Auto-Regressive Decoding
PL/SQL	Procedural Language/Structured Query Language
RAG	Retrieval-Augmented Generation

Abbreviation	Full Form
REST	Representational State Transfer
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
RQ	Research Question
SH	Sales History (Oracle sample schema)
SQL	Structured Query Language
SUS	System Usability Scale
ToTTo	Towards Totally Trustworthy table-to-text (dataset)
USD	United States Dollar
VLDB	Very Large Data Bases (conference)
WJARR	World Journal of Advanced Research and Reviews
WJAETS	World Journal of Advanced Engineering Technology and Sciences

Chapter 1 Introduction

1.1 Overview

Today organizations are creating and storing more data than ever before. Relational databases are the backbone of daily business operations, storing transactions, employee records, customer information, inventory data and financial activities. But although the amount of data available is increasing, decision makers still struggle to access and interpret it efficiently. In many organizations, a new business question requires technical expertise and developer support to answer, slowing down and making the reporting process less flexible.

Organizations have invested heavily in business intelligence (BI) and analytics technologies to improve data access and decision-making. The global BI and analytics market was valued at approximately USD 54.9 billion in 2025 and is expected to grow at a compound annual growth rate of 12.4%.

Despite these investments, many organizations still struggle to make data easily accessible to decision-makers. Research shows that only 32% of executives feel confident using data effectively (G2,2024). In addition, data preparation accounts for approximately 45% of data scientists' time before analysis can begin (ALTEN, 2025).

Even among organizations that have adopted BI platforms, self-service analytics has reduced report generation backlogs by approximately 40%, demonstrating its potential to reduce reliance on developers and improve access to organizational data (DataStackHub, 2025).

Language is another challenge, particularly in the MENA region. Most BI and reporting tools are designed for English, making it difficult for Arabic-speaking users to interact with data in their native language. This is especially important

because many organizations in the Gulf region use Oracle Database and require administrative reports in Arabic.

Oracle Application Express (APEX) provides a suitable platform to address this problem. It is a low-code development platform that runs on Oracle Database and is widely used to build enterprise web applications. Because APEX is tightly integrated with the database, it provides a suitable environment for integrating natural language querying and intelligent reporting. However, as discussed in Chapter 2, limited research has explored the integration of these capabilities within Oracle APEX. This study addresses this gap by proposing an NLP-based reporting framework that enables users to generate reports through natural language.

1.2 Problem Statement

In most organizations, decision-makers face obstacles when they need data-driven answers; every time they want them, they need to get them from the developer.

Enterprise reporting systems depend on fixed dashboards and pre-designed queries. When administrators need additional insights from the data, such as quarter-to-quarter comparison, departmental analysis, or explanations for unusual trends, they need to request a new report and wait for the developer to design it and deploy it. After the report is generated, administrators still need to interpret rows of numbers on their own, often without any explanation or context. This process can take days or even weeks depending on the developer's availability and the number of pending requests.

The current reporting process suffers from three core challenges. First, the reliance on developers to produce reports means that executives can only make decisions as fast as developers are able to create the requested report. Second, existing reports show row data in tables and charts without clear explanations, identification of unusual patterns, or highlighting of the most important insights. Third, most NLP tools are for English rather than Arabic. This implies that Arabic-speaking executives have poor access to natural language query features and thus cannot easily interact with company data in their native language.

This research addresses these three challenges by proposing an NLP-based layer integrated into Oracle APEX. The proposed solution allows decision-makers to ask a question in Arabic or English, automatically converts it into a validated SQL query, executes it on the organization's database, and returns the result with a brief explanation of the data.

The goal is to reduce the need for the developer by allowing the decision-makers to interact directly with the system that is fast, faithful, and bilingual.

1.3 Research Questions

This research is motivated by three questions: How accurate is the query component? How faithful is the reporting component? And what architecture is needed to make the two function securely inside APEX?

- ***RQ1:*** *How effective is a prompt-engineered large language model at translating Arabic and English natural-language questions to correct SQL queries against enterprise Oracle databases?*

- **RQ2:** *How effective are LLM-based data-to-text techniques at generating accurate and readable narrative summaries from SQL query results in an administrative reporting context?*
- **RQ3:** *What architectural components are needed to integrate NL2SQL and natural language generation securely and effectively within Oracle APEX?*

RQ1 and RQ2 measure the performance of the NL2SQL and NLG components. RQ3 is concerned with the integration of these components in Oracle APEX such as schema linking, security, SQL validation and the user interface.

1.4 Significance of the Study

This study contributes to research and practice in the areas of Natural Language Processing (NLP), enterprise information systems, and low-code application development.

A theoretical contribution of this study is the application of Natural Language to SQL (NL2SQL) and data-to-text generation in a bilingual (Arabic–English) enterprise environment. Most previous studies have focused on English benchmarks or general-purpose datasets, whereas this study applies these technologies to administrative reporting, where both Arabic and English are required.

The study also has practical value. It gives administrators and decision-makers a way to create reports by asking questions in natural language instead of writing SQL queries or requesting new reports from developers. The system generates the

SQL query and then produces a short narrative summary of the results. This can save time, improve the reporting process, and help decision-makers obtain the information they need more quickly.

Another contribution of this study is the bilingual evaluation framework. It includes administrative questions in Arabic and English together with their corresponding SQL queries and generated reports. The framework is used to evaluate the performance of both NL2SQL and Natural Language Generation (NLG) in an enterprise reporting environment. It may also be useful for future research on multilingual intelligent reporting.

The study also proposes a secure and modular architecture for Oracle APEX. The architecture combines Large Language Models (LLMs), prompt engineering, SQL validation, and report generation in a single framework. It shows one way of integrating these technologies into Oracle APEX and can be used as a reference for future AI-enabled reporting applications built on Oracle technologies.

1.5 Scope and Limitations

This study focuses on developing an NLP-based reporting system within Oracle APEX. The work is limited to an Oracle database using the standard HR and Sales History schemas, which provide structured relational data with well-defined tables, columns, and relationships. Other areas of enterprise reporting are outside the scope of this study.

Only read-only SQL queries are supported, so the database cannot be modified by the system. Users can ask questions in either Modern Standard Arabic or English, and the generated reports are produced in the same language as the user's request.

The NLP part of the system performs two main tasks. It translates natural language questions into SQL queries and generates brief summaries to explain the query results. These tasks rely on prompt engineering and in-context learning with large language models, and no model fine-tuning is carried out.

Before a generated SQL query is executed, it is displayed to the user together with a plain-language explanation. The user can review the query before deciding to run it. Access to data follows the permissions already defined in Oracle APEX, so users can only view information they are authorized to access.

The limitations of the study are as follows:

- The proposed system is Oracle database specific and does not support other database platforms like Postgres or SQL Server. It is limited to HR and Sales History schemas used in this study.
- The system is designed to work with the structured data stored in the Oracle database. It does not support real-time streaming or unstructured data, and all write operations are disabled to protect the database.
- The system supports only Modern Standard Arabic and English. Arabic dialect and other languages are beyond the scope of the study.
- The user interface is a text-based interface and does not support voice input.
- The prototype handles one-turn queries. It does not remember conversation context from one turn to the next or learn over time from user interactions.
- The system relies on Oracle APEX's built-in security framework. Advanced security features are outside the scope of this study because

the prototype uses Oracle's sample schemas instead of a real organization's data.

- User evaluation is conducted with a small group of participants to gather qualitative feedback on the proposed system.

These limitations were intentionally defined to keep the study feasible and to ensure that the results accurately represent what the proposed system can and cannot do.

1.6 Approach

The input formats collected during the data phase include a customized multilingual assessment set and the general criteria used for testing, and the organizational database that the system reports on, which was created using Oracle's human resources and sales history charts. The setup process standardizes the Arabic text format and removes typographical variations that do not alter the meaning to make the text easier to process. It also gathers chart metadata from the Oracle Data Dictionary and creates a vector search retrieval store so that relevant dates and terms can be entered into each query at query time.

The build process creates an NL2SQL component, which develops and validates the SQL through human confirmation before execution, and a narrative build component, which transforms query results into a concise and accurate description. PL/SQL and REST are used for integration in the APEX application.

During the evaluation phase, narrative accuracy is measured using automated metrics and human review, and SQL accuracy is tested using standards and a

bilingual suite. Task-based user research provides qualitative feedback on the system. Technical details for each step are provided in Chapter 3.

1.7 Thesis Organization

This thesis is organized into six chapters that move the reader from the initial identification of the problem through to the final conclusions.

1 Introduction — This chapter introduces the research context, including the problem of inaccessible ad-hoc data querying in Oracle APEX environments and the proposed solution, an NLP layer that maps natural-language questions to SQL and generates narrative summaries. The chapter states the research questions, describes the research significance, and presents the scope and limitations of the research.

Chapter 2 — Literature Review — examines previous research in the areas upon which this thesis is based. It reviews the development of data conversion to SQL, its main criteria and limitations, and the challenges of generating high-precision text. It also reviews recent literature on integrating artificial intelligence into low-code programming platforms, specifically Oracle APEX, and concludes by identifying the research gap that this thesis will address: there is currently no study that combines data conversion to accurate SQL, accurate natural language generation, and the ability to use two languages in an APEX reporting application.

Chapter 3 — Methodology — describes the technical blueprint for building the system. It covers Oracle database and dataset setup, data preparation for model readability, selection and design of NL2SQL and NLG modules, and the evaluation framework including building a bilingual test suite and SQL accuracy and narrative accuracy evaluation metrics.

Chapter 4 — System Implementation — covers the practical implementation of the prototype, from setting up the Oracle APEX application to creating NL2SQL and NLG modules with rapid engineering and safety controls. It reviews the complete user flow from query to narrative summary and describes the configuration of Oracle's native AI features – AI identification and the APEX AI assistant – as key comparisons.

Chapter 5 — Evaluation and Results — presents the empirical results. It shows the accuracy of SQL queries on public benchmarks and a custom bilingual test set, evaluates narrative quality using both automated metrics and human evaluation, and compares the proposed system to Oracle's built-in AI features. The chapter includes qualitative observations from end users, and discusses the findings in light of each research question.

Chapter 6 — Conclusion and Future Work — brings together elements of the thesis. This chapter discusses contributions, lessons learned during the development process, limitations of the prototype, and identifies directions for future work, such as expanding schema support, enabling multi-role conversations, and developing a ready-to-use version of the system.

1.8 Expected Results

By the end of the project, we primarily expect to provide a powerful tool within Oracle APEX that allows administrators to type any question in Arabic or English. The system will then generate the correct query, execute it, and display the results along with a concise narrative summary.

We also anticipate conducting a thorough assessment of the query generation efficiency on both the public and private test sets for Arabic and English, as well as measuring the accuracy and reliability of the resulting written summaries.

The project should also enable useful comparisons. We want to see the differences between Oracle AI and our approach, particularly in Arabic, given its limited support.

Finally, the project aims to leave a lasting impact by providing a clear roadmap for integrating this type of natural language reporting into other APEX applications, creating a bilingual (Arabic and English) test set for administrative questions that did not previously exist, and offering practical lessons on maintaining system accuracy and security.

1.9 Resources Needed

1. The Oracle platform according to Oracle (2024) comprises Oracle Database 23ai or Autonomous Database (OCI Always Free tier is enough for development) and Oracle APEX (apex.oracle.com or a local version).
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. Datasets: Spider, BIRD, Spider 2.0; ToTTo, DART, WebNLG; Oracle sample schemas; plus the custom bilingual set.
4. Development tools: VS Code, SQL Developer / SQLcl, Python (transformers, datasets, evaluation libraries), Git.

Chapter 2 Background and Related Work

2.1 Background

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 Question Answering for NL2SQL

This problem is usually raised from a translation perspective: when a query is given in natural language and the database schema is provided, an SQL command is

generated, executed, and produces the correct answer. A good model produces a grammatically incorrect SQL statement, prompting early attempts to restrict the output. In the PICARD project, Sholac, Shuscher, and Behdano (2021) restricted the generation process by adding only the correct characters at each decoding step, thus reducing syntax errors in the generated SQL without requiring the model to be retrained.

Later, the rise of LLMs changed the paradigm. Instead of training a dedicated parser, researchers showed that a general model, when provided with the schema and some examples, was able to produce SQL statements with prompts. Pourreza and Rafiei (2023), in DIN-SQL, showed that decomposing the problem into several stages and allowing the model to inspect its own draft resulted in better results than plain prompting. Gao et al. (2024), with DAIL-SQL, opted for a reflective approach: analyzing which prompts were effective by comparing the representation of the user query and schema and by analyzing the choice and ordering of in-context examples. This work is more about describing the effective methodologies than about introducing a new model.

Benchmarking has become more realistic over time. In the BIRD benchmark (Li et al., 2023), large but flawed database schemas were created, which needed some additional data, and the benchmarking covered not only the efficiency but also the correctness of the answer, hence identifying significant deficiencies in the top models compared to human specialists. In Spider 2.0 (Lei et al., 2025), the benchmarking involved real-world complexity of schema and analysis in professional context, where contemporary models can solve only part of those problems. Recent two surveys, Hong et al. (2024) and Shi et al. (2024), which

describe the rapidly changing terrain by the same axis: prompt-based models and fine-tuning models.

Looking at this whole picture together, there are a few things that remain unclear for anyone designing an admin reporting tool. Benchmarks are mostly English and generic, so good performance says nothing about a particular business' tables and terminology. It is still not clear how to handle enterprise-scale messy data, as is shown by BIRD and Spider 2.0 results. The security issues related to the generated query, including its read-only capability, proper permissions and pre-execution validation, are often ignored by researchers although they define the very possibility of implementation. The bilingual case, i.e., Arabic and English, is hardly addressed in the literature.

2.3 Turning data into written reports: text generation

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.

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

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.

In another study Keshireddy (2025b) moves from interaction to data integrity in the Journal of Internet Services and Information Security (JISIS). This paper introduced an adaptive AI framework in Oracle APEX that automates data transformation and validation based on semantic, statistical and constraint-based techniques rather than the inflexible PL/SQL rules that most APEX systems utilize. It was validated on 980,000 records in HRMS, CRM and Accounting modules with transformation accuracy above 92% in HRMS and CRM and 94.6% in Accounting. The framework also offers encryption, access control and audit logging, and adapts automatically when schemas change. This relates to two objectives in this thesis, the need to deal reliably with real enterprise data and the need to make sure that the operations of the system are secure and auditable. It is the most evidence-based 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.

2.6 A Worked Example: From Question to Report

We will list an example for the proposed full life cycle of this model, including the main three sections: How the system will translate questions to SQL queries, how the text will be generated from the returned data, and how we will implement it in low-code platform APEX.

We will be using Oracle's Sales History (SH) schema, which contains:

transactional sales records, product information, customer data, promotional campaigns, and time dimensions.

The suggested question is a multi-level query which is complex enough and goes thru different tables, compared by years, containing derived and calculated columns.

The full pipeline is shown in Figure 1. We provided an explanation for each step and figure 2 is an example for the cycle from start (user input) to end (report and data display).

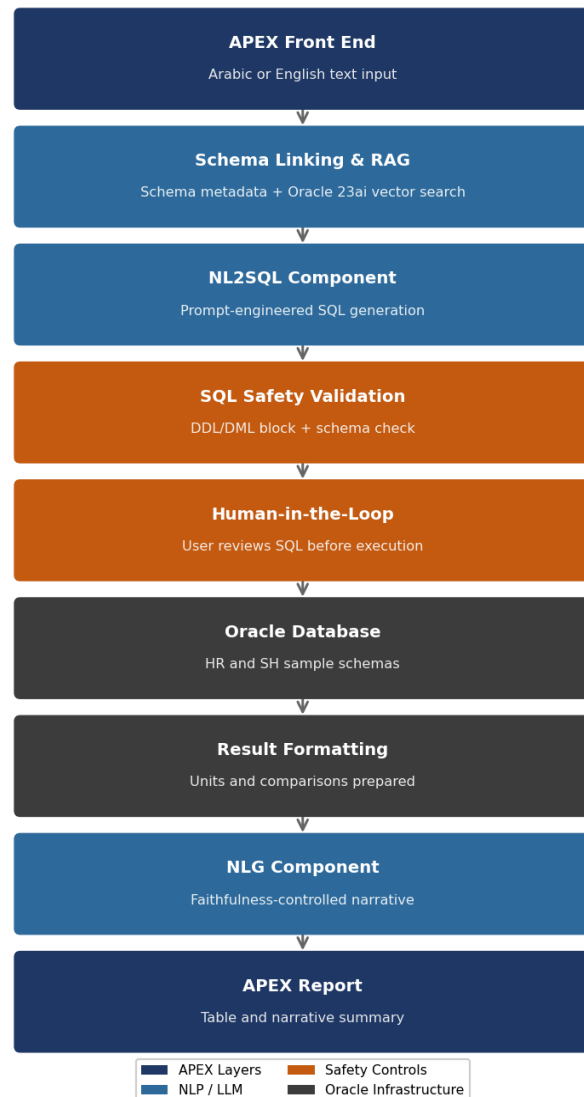


Figure 1. 2.6 System pipeline architecture for the proposed NLP-based reporting framework

2.6.1 The Scenario

A sales director wants to understand which product categories grew in sales in 2001 compared to the year before, and whether the company's promotional discounting strategy contributed to this growth. Under the current setup, she would need to file a report request and wait for a developer to build a new query across multiple tables, then build a new web page to show the results. With the proposed system, she types her question directly into the APEX interface.

2.6.1.1 User Input

The user types her question in English:

"Which product categories had the highest revenue growth in 2001 compared to 2000, and what was the average promotional discount for each category?"

Or the equivalent in Arabic:

ما هي أنواع المنتجات التي حققت أعلى مبيعات خلال عام 2001 مقارنةً بعام 2000، وما متوسط نسبة الخصومات لهذه الفئات؟

The language will be auto-detected. Depending on the language there will be processed. Arabic needs all the following:

- Normalization step standardizes *alef* variants
- Removing diacritics
- Handling *ta marbuta* endings before any further processing.

While the English needs no processing and passes through unchanged.

2.6.1.2 Schema Linking and Context Retrieval

This question needs many tables to get the requested results and it is not as easy as retrieving data from single table. It needs to access SH's data dictionary to identify the relevant tables: **SALES** (containing *AMOUNT_SOLD*, *QUANTITY_SOLD*, *PROD_ID*, *TIME_ID*, *PROMO_ID*), **PRODUCTS** (*PROD_ID*, *PROD_CATEGORY*, *PROD_LIST_PRICE*), **TIMES** (*TIME_ID*, *CALENDAR_YEAR*), and **PROMOTIONS** (*PROMO_ID*, *PROMO_CATEGORY*). It must maps the phrase "revenue growth" to a year-over-year difference across *AMOUNT_SOLD* grouped by *CALENDAR_YEAR*, and maps "promotional discount" to a derived new column comparing *PROD_LIST_PRICE* against the average sale price on promotion-tagged rows.

2.6.1.3 NL2SQL Generation

The assembled prompt goes to the large language model. The model reasons step by step: it identifies the relevant tables and joins, notes that the growth calculation requires conditional aggregation by year rather than a subquery, chooses *LEFT JOIN* for promotions since some sales carry no promotional code, and adds *NULLIF* to guard against division by zero in the growth percentage formula. The output is:

Sql

Table 1 2.6 NL2SQL Query

```
SELECT

    p.prod_category,

    ROUND(SUM(CASE WHEN t.calendar_year = 2000

        THEN s.amount_sold ELSE 0 END), 2) AS revenue_2000,

    ROUND(SUM(CASE WHEN t.calendar_year = 2001
```

```

        THEN s.amount_sold ELSE 0 END), 2) AS revenue_2001,

ROUND(

    (SUM(CASE WHEN t.calendar_year = 2001 THEN s.amount_sold ELSE 0 END)

    - SUM(CASE WHEN t.calendar_year = 2000 THEN s.amount_sold ELSE 0 END))

    / NULLIF(SUM(CASE WHEN t.calendar_year = 2000

        THEN s.amount_sold ELSE 0 END), 0) * 100, 2

    ) AS growth_pct,

ROUND(AVG(

    CASE WHEN s.promo_id != 999

        THEN (p.prod_list_price

            - s.amount_sold / NULLIF(s.quantity_sold, 0))

            / NULLIF(p.prod_list_price, 0) * 100

        ELSE NULL END

    ), 2) AS avg_promo_discount_pct

FROM sales s

JOIN products p ON s.prod_id = p.prod_id

JOIN times t ON s.time_id = t.time_id

LEFT JOIN promotions pr ON s.promo_id = pr.promo_id

WHERE t.calendar_year IN (2000, 2001)

GROUP BY p.prod_category

ORDER BY growth_pct DESC

```

This query is more complex than a typical report query. It has conditional aggregation, a derived column with zero-division protection, a per-row discount calculation averaged across promotion-tagged sales, a LEFT JOIN for optional promotion data. Alos, there is safety measures to make sure no DDL or DML keywords, no dynamic SQL, and that every table and column reference exists in the current schema.

2.6.1.4 Human Confirmation

Rather than running the query immediately, the system presents it to the user alongside a plain-language explanation: "This query compares revenue across 2000 and 2001 for each product category, calculates the year-on-year percentage change, and adds the average discount applied during promotional sales. It reads from four tables and will not change any data." The user reviews this, judges it accurate, and confirms. Only at that point is the query executed. It is crucial to have this step in the early stages of implementation and testing to ensure all queries are applicable and perform as the user wishes.

2.6.1.5 Execution and Result Formatting

Upon running the query, five rows are returned, one per product category. We could use some formatting for better understanding such as adding currency symbols, rounding percentages, and structuring the output before passing it to the narrative generator. Table 2 shows the results.

Table 2 2.6 Execution and Result Formatting

Product Category	Revenue 2000 (USD)	Revenue 2001 (USD)	Growth (%)	Avg Promo Discount (%)
Software/Other	\$9,124,200	\$11,456,800	+25.6	18.4
Electronics	\$8,345,600	\$9,876,300	+18.3	14.2

Hardware	\$6,782,100	\$7,432,900	+9.6	12.7
Peripherals & Accessories	\$4,123,400	\$4,398,200	+6.7	15.9
Photo	\$2,456,700	\$2,198,500	−10.5	22.1

Table 2. Query results: revenue growth by product category, 2000 vs 2001 (Oracle SH schema)

2.6.1.6 Narrative Generation

NLG component receives three different values: the asked question, the returned data and the auto detected language. After that a prompt will be fired that explains to the component the data fully, its columns and the meaning behind each cell to avoid hallucination. This will be an example of an english result:

"Software/Other recorded the strongest year-on-year revenue growth among all product categories at 25.6%, adding approximately \$2.3 million between 2000 and 2001, with an average promotional discount of 18.4%. Electronics followed at 18.3% growth on a 14.2% average promotional discount. Hardware and Peripherals and Accessories grew at single-digit rates of 9.6% and 6.7%, respectively. Photo was the only category to report a decline, falling 10.5% year-on-year while carrying the highest average promotional discount of any category at 22.1%."

Same goes for Arabic; it follows the same validations and restrictions.

2.6.1.7 Report Display

The APEX report page shows both the table from Step 5 and the narrative from Step 6 together. The sales director has her answer in seconds: two categories grew strongly, two grew modestly, and one contracted — all with the promotional discount context she asked for — without writing SQL or filing a request. If she

wants to go deeper into a single category or a specific region, she types the next question and the cycle begins again from Step 1.

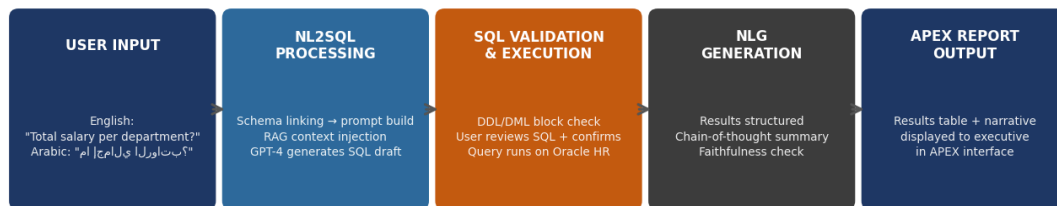


Figure 2 2.6 Complete cycle from user input to executive report for the worked example scenario

This example illustrates the three research questions. RQ1 is answered by Step 3, where the system generates a structurally complex SQL statement — conditional aggregation, a derived metric, different types of JOIN, and NULLIF — from a natural-language question against a real Oracle schema. RQ2 is answered by Step 6, where the narrative stays faithful to the table, covers all five categories in order, and avoids going beyond the data. RQ3 is answered by the full pipeline in Figure 1, which shows the architectural components required to handle this level of query complexity safely.

Appendix: Synthesis Matrix

Table 3 2.5 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

Work	Area	What it does	Language	Limitation for this thesis
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
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

Work	Area	What it does	Language	Limitation for this thesis
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

[ALTEN, 2025]

ALTEN. (2025, April 25). 2024 universal registration document and the integrated report including the annual financial report. ALTEN SA. https://www.alten.com/wp-content/uploads/2025/05/2024_Universal_Registrati on_Document.pdf

[Gao, D., et al., 2024]

Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., & Zhou, J. “Text-to-SQL empowered by large language models: A benchmark evaluation”. *Proceedings of the VLDB Endowment*, 17(5), 1132–1145, 2024. <https://arxiv.org/abs/2308.05389>

[DataStackHub, 2025]

DataStackHub. (2025, October 25). Business intelligence (BI) statistics for 2025–2026. Data Stack Hub. <https://www.datastackhub.com/insights/business-intelligence-statistics/>

[G2, 2024]

G2. (2024). 35+ business intelligence statistics for 2025. G2. <https://www.g2.com/articles/business-intelligence-statistics>

[Hong, Z., et al., 2024]

Hong, Z., Yuan, Z., Zhang, Q., Chen, H., Dong, J., Huang, F., & Huang, X. “Next-generation database interfaces: A survey of LLM-based text-to-SQL”. *arXiv*, 2024. <https://arxiv.org/abs/2406.08426>

[Ji, Z., et al., 2023]

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. “Survey of hallucination in natural language generation”. *ACM Computing Surveys*, 55(12), 1–38, 2023. <https://arxiv.org/abs/2202.03629>

[Keshireddy, S. R., 2025b]

Keshireddy, S. R. “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, 2025. <https://doi.org/10.58346/JISIS.2025.12.041>

[Keshireddy, S. R., 2025a]

Keshireddy, S. R. “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), 2025.

[Keshireddy, S. R., 2025d]

Keshireddy, S. R. “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), 2025.

[Keshireddy, S. R., 2025c]

Keshireddy, S. R. (2025c). “Low-code development enhancement integrating large language models for intelligent code assistance in Oracle APEX”. 2026. <https://orcid.org/0009-0007-6482-4438>

[Lei, F., et al., 2025]

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. “Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows”. *International Conference on Learning Representations (ICLR)*, 2025. <https://arxiv.org/abs/2411.07763>

[Li, J., et al., 2023]

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. “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, 2023. <https://arxiv.org/abs/2305.03111>

[Li, W., et al., 2022]

Li, W., Wu, W., Chen, M., Liu, J., Xiao, X., & Wu, H. “Faithfulness in natural language generation: A systematic survey of analysis, evaluation and optimization methods”. *arXiv*, 2022. <https://arxiv.org/abs/2203.05227>

[Lin, Y., et al., 2024]

Lin, Y., Ruan, T., Liu, J., & Wang, H. “A survey on neural data-to-text generation”. *IEEE Transactions on Knowledge and Data Engineering*, 36(4), 1431–1449, 2024. <https://doi.org/10.1109/TKDE.2023.3304385>

[Nan, L., et al., 2021]

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. “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, 2021.
<https://arxiv.org/abs/2007.02871>

[Oracle Corporation, 2024a]

Oracle Corporation. (2024). Announcing Oracle APEX 24.2 general availability: Generative development, AI configurations, and RAG sources [Blog post]. Oracle APEX Blog. <https://blogs.oracle.com/apex/post/oracle-apex-242-is-generally-available>

[Oracle Corporation, 2024b]

Oracle Corporation. (2024). Configuring generative AI in Oracle APEX using Select AI and DBMS_CLOUD_AI [Technical documentation]. Oracle Corporation. <https://docs.oracle.com/en/database/oracle/oracle-database/23/aexml/>

[Oracle Corporation, n.d.]

Oracle Corporation. (n.d.). Oracle APEX AI platform [Product information]. Oracle Corporation. <https://apex.oracle.com/en/platform/ai/>

[Oyewole, A. T., et al., 2024]

Oyewole, A. T., Adeoye, O. B., Addy, W. A., Okoye, C. C., Ofodile, O. C., & Ugochukwu, C. E. “Automating financial reporting with natural language processing: A review and case analysis”. *World Journal of Advanced Research and Reviews*, 21(3), 575–589, 2024. <https://doi.org/10.30574/wjarr.2024.21.3.0688>

[Parikh, A. P., et al., 2020]

Parikh, A. P., Wang, X., Gehrmann, S., Faruqui, M., Dhingra, B., Yang, D., & Das, D. “ToTTo: A controlled table-to-text generation dataset”. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1173–1186, 2020. <https://arxiv.org/abs/2004.14373>

[Pourreza, M., et al., 2023]

Pourreza, M., & Rafiei, D. “DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction”. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023. <https://arxiv.org/abs/2304.11015>

[Rao, N., et al., 2024]

Rao, N., Tsay, J., Kate, K., Hellendoorn, V. J., & Hirzel, M. “AI for low-code for AI”. *Proceedings of the 29th International Conference on Intelligent User Interfaces (IUI '24)*, 837–852, 2024. <https://doi.org/10.1145/3640543.3645203>

[Roychowdhury, S., et al., 2024]

Roychowdhury, S., Krema, M., Mahammad, A., Moore, B., Mukherjee, A., & Prakashchandra, P. “ERATTA: Extreme RAG for table-to-answers with large language models”. *IEEE International Conference on Big Data*, 2024. <https://arxiv.org/abs/2405.03963>

[Scholak, T., et al., 2021]

Scholak, T., Schucher, N., & Bahdanau, D. “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, 2021. <https://arxiv.org/abs/2109.05093>

[Sharma, A., et al., 2025]

Sharma, A., Giri, S., Pandey, S., Pandey, P., & Pandey, S. K. “A study on natural language processing techniques for intelligent text generation”. *International Journal for Research in Applied Science & Engineering Technology*, 13(12), 2025. <https://doi.org/10.22214/ijraset.2025.76052>

[Shi, L., et al., 2024]

Shi, L., Tang, Z., Zhang, N., Zhang, X., & Yang, Z. “A survey on employing large language models for text-to-SQL tasks”. *arXiv*, 2024. <https://arxiv.org/abs/2407.15186>

[Syed, A., 2025]

Syed, A. “Leveraging generative AI in Oracle APEX for intelligent applications [Preprint]”. *ResearchGate*, 2025. <https://www.researchgate.net/publication/397555550>

[Yagamurthy, D. N., et al., 2023]

Yagamurthy, D. N., Azmeera, R., & Khanna, R. “Natural language generation (NLG) for automated report generation”. *Journal of Technology and Systems*, 5(1), 48–59, 2023. <https://doi.org/10.47941/jts.1497>

[Yeruva, S. K., 2025]

Yeruva, S. K. “Leveraging AI features in Oracle APEX for smart application development”. *World Journal of Advanced Engineering Technology and Sciences*, 16(1), 624–633, 2025. <https://doi.org/10.30574/wjaets.2025.16.1.1239>

[Zhao, Y., et al., 2023]

Zhao, Y., Zhang, H., Si, S., Nan, L., Tang, X., & Cohan, A. “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, 2023. <https://arxiv.org/abs/2305.14987>

