



# **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

## 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

Enterprise and government organizations still rely on static, developer-built dashboards for administrative and executive reporting that are not responsive to ad-hoc questions and do not provide a narrative explanation of the data presented. This paper describes the architecture and design of a bilingual natural language processing framework embedded in Oracle Application Express (APEX) that overcomes both of these limitations.

The proposed system can accept queries in Arabic or English, translate them into validated, read-only SQL with prompt-engineered large language model, augmented by retrieval-augmented generation and a human-in-the-loop confirmation step, execute the query, and generate a concise, faithful narrative summary by a second language model stage.

This framework will be deployed on Oracle APEX using one of Oracle's schemas: Sales History schema, HR Schema or products\_order schema. The architecture includes: schema linking, RAG pipeline, safety layer to validate SQL queries, narrative generation component with controlled faithfulness and minimum hallucination which supports both Arabic and English language.

In this paper we will present the full architecture for our framework that covers an accurate NL2SQL, faithful narrative and efficient user reports, identifying the research gap this paper addresses: no current study presents a validated NL2SQL and controlled NLG in a bilingual administrative reporting tool within a low-code enterprise platform that supports both standard Arabic and English. 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         |
| <b>Chapter 1 Introduction</b>                           | <b>10</b> |
| 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        |
| <b>Appendix: Synthesis Matrix</b>                       | <b>34</b> |
| References                                              | 37        |

## List of Tables

|                                             |    |
|---------------------------------------------|----|
| Table 2.6.1 NL2SQL Query                    | 29 |
| Table 2.6.2 Execution and Result Formatting | 31 |
| Table 2.5.3 Synthesis Matrix                | 34 |

## List of Figures

|                                                                                                 |    |
|-------------------------------------------------------------------------------------------------|----|
| Figure 2.6.1 System pipeline architecture for the proposed NLP-based reporting framework        | 26 |
| Figure 2.6.2 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                                               |
|--------------|---------------------------------------------------------|
| <b>ACM</b>   | Association for Computing Machinery                     |
| <b>AI</b>    | Artificial Intelligence                                 |
| <b>API</b>   | Application Programming Interface                       |
| <b>APEX</b>  | Application Express (Oracle)                            |
| <b>BI</b>    | Business Intelligence                                   |
| <b>BIRD</b>  | Big Bench for Large-Scale Database Grounded Text-to-SQL |
| <b>BLEU</b>  | Bilingual Evaluation Understudy                         |
| <b>CAGR</b>  | Compound Annual Growth Rate                             |
| <b>CRM</b>   | Customer Relationship Management                        |
| <b>DART</b>  | Open-Domain Structured Data Record to Text (dataset)    |
| <b>DDL</b>   | Data Definition Language                                |
| <b>DML</b>   | Data Manipulation Language                              |

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

| Abbreviation  | Full Form                                                     |
|---------------|---------------------------------------------------------------|
| <b>REST</b>   | Representational State Transfer                               |
| <b>ROUGE</b>  | Recall-Oriented Understudy for Gisting Evaluation             |
| <b>RQ</b>     | Research Question                                             |
| <b>SH</b>     | Sales History (Oracle sample schema)                          |
| <b>SQL</b>    | Structured Query Language                                     |
| <b>SUS</b>    | System Usability Scale                                        |
| <b>ToTTo</b>  | Table-To-Text (controlled generation dataset)                 |
| <b>USD</b>    | United States Dollar                                          |
| <b>VLDB</b>   | Very Large Data Bases (conference)                            |
| <b>WJARR</b>  | World Journal of Advanced Research and Reviews                |
| <b>WJAETS</b> | 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. Although the amount of available data 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 the process and making reporting 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 has both research and practical contributions in Natural Language Processing (NLP), enterprise information systems, and low-code application development. It applies 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, while this study focuses on administrative reporting where both Arabic and English are needed.

The study also has a practical benefit for administrators and decision-makers. It allows them to create reports by asking questions in natural language instead of writing SQL queries or asking developers to create new reports. The system generates the SQL query and then provides a short summary of the results. This can save time, make the reporting process easier, and help decision-makers get the information they need faster.

Another contribution of this study is the bilingual evaluation framework. It includes administrative questions in Arabic and English, their SQL queries, and the generated reports. The framework is used to evaluate NL2SQL and Natural Language Generation (NLG) in an enterprise reporting environment. It can also help with future research on multilingual reporting.

The study also proposes a secure and modular architecture for Oracle APEX. It combines Large Language Models (LLMs), prompt engineering, SQL validation,

and report generation in one system. This shows how these technologies can be used together in Oracle APEX and can be useful for future reporting applications based on Oracle technologies.

## **1.5 Scope and Limitations**

This paper aims to conceptualize an NLP-based reporting system in Oracle APEX. However, the system only uses Oracle databases under HR and Sales History schemas; thus, the system cannot be reported for a general enterprise due to different concepts of relationally stored data and the arrangement of tables, columns, and their relationships conceptually different from that of these sample schemas employed in this research project.

The system supports only Structured Query Language (SQL) queries in read-only mode; thus, it will not be able to update or insert data into the database. From Modern Standard Arabic (MSA) or English, the system can respond to users' questions in either language. The reports will thus be generated in the default language specified for a particular user profile.

The system performs two key NLP system functions: translating questions into Structured Query Language (SQL) queries and summarizing the results obtained from query execution in plain language. This approach is based on invoking prompt engineering and in-context learning of large language models, with no fine-tuning required.

The system first displays any generated Structured Query Language (SQL) query in natural language and will only display the query to users for execution. Therefore, the preceding execution allows the user to intercept the development and execution

of the query. Access to data is thus restricted, according to the models, business logic, and the rights assigned in Oracle APEX.

This study's limitations include specific functionalities discussed related to databases, as they deal exclusively with the Oracle database and schemas under HR and Sales History.

The suggested framework is available only within Oracle databases and cannot be embedded within other database engines such as Postgres or Structured Query Language (SQL) Server.

Moreover, the current study, rather than focusing on this, deals with structured data from Oracle database management. Databases rely on the fact that continuous streaming and unstructured data are not supported. All writing operations are temporarily suspended, preserving the database's integrity.

This paper considers Arabic in its modern standard dialect and, hence, does not account for varieties or other relevant languages.

This text-based system cannot build a voice-based interface.

Multi-turn queries, referring to more than one query, have not been realized in the context of conversation or argument by the system prototype.

These are all very small-sample qualitative user evaluations of some critical aspects of the system.

Finally, these constraints founded this research on a reasonable basis to make well-grounded propositions on the actual system's capabilities in the entire space.

## 1.6 Approach

The data acquired was the customized multilingual test set of general test standards and organizational information that had been captured while setting up the designed charts of human resources and sales history given by Oracle and reported from databases through a single inquiry. During preprocessing, the standardization of Arabic text and the elimination of spelling differences that do not make semantic and syntactic changes to the text simplify processing. Accordingly, the chart metadata in the Oracle Data Dictionary is obtained, and vector search retrieval storage of all relevant dates and names during querying is requested.

Then, an NL2SQL module will be generated to establish the development and validation of SQL after user confirmation before actual execution of the final stage of integration with the APEX application further requiring PL/SQL and REST. Besides that, a narrative module is related to transforming query results into brief and meaningful descriptions.

This is followed by measures of the accuracy of narratives with automatic measures and human review, while correctness-oriented SQL is tested using standards and a bilingual suite. Chapter 3 describes the process in detail per each step.

## 1.7 Thesis Organization

This thesis is outlined in six chapters, the first outlining a problem and finally drawing 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 — explains how the system is designed and developed. It covers the setup of the Oracle database and datasets, preparing the data for the model, and selecting and designing the NL2SQL and NLG modules. It also explains how the system is evaluated, including the bilingual test set and the measures used to assess SQL and narrative accuracy.

Chapter 4 — System Implementation — describes how the prototype was implemented in Oracle APEX. It covers the setup of the APEX application, the development of the NL2SQL and NLG modules, and the safety controls used in the system. It also explains the complete user process, starting from entering a question and ending with the generated narrative summary. The chapter also describes the setup of Oracle’s built-in AI features, including Select AI and the APEX AI Assistant, which are used to compare their performance with the proposed system.

Chapter 5 — Evaluation and Results — presents the results of the evaluation. It examines SQL query accuracy using public benchmarks and a custom bilingual test

set. It also evaluates the quality of the generated narratives using automated measures and human evaluation. The proposed system is compared with Oracle's built-in AI features, and feedback from end users is also discussed. The results are then discussed in relation to each research question.

Chapter 6 — Conclusion and Future Work — summarizes the main findings of the thesis. It discusses the study's contributions, what was learned during the development of the system, and the main limitations of the prototype. It also suggests some areas for future work, such as adding support for more schemas, enabling multi-role conversations, and developing a ready-to-use version of the system.

## **1.8 Expected Results**

By the end of this project, we hope to get a good product of Oracle APEX, which allows the administrator to type any question in Arabic or English, get, and run the appropriate query, then present the results and summarize them in a brief narrative.

Moreover, detailed performance reviews will measure the public and private test datasets in both Arabic and English and their coverage throughout the query generation process. There will also be coverage performance tests on the validity and reliability of the generated summarization process.

Furthermore, this should be quite the opposite result in which valuable comparative results will be obtained between Oracle AI and the present work, especially in Arabic, where little support is available.

Finally, the project will be beneficial in sustaining a practical line to bring this kind of natural language reporting into other APEX applications besides the first

bilingual test dataset in Arabic and English for administrative questions and experience in cases of such systems' post-deployment 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

Data concerning everything an organization does is part of the business. Data about finance, human resources, operational issues, and even the daily running of support cases sits in any typical RDBMS. Briefly, however, pertinent such information is, data is not anything close to sufficient. Many variables determine how accurately the data can be represented when the results are given either quickly or in a way understandable by a manager without further investment in software development to sustain reporting. Therefore, this thesis logically stands where the two sides of the equation meet-the question asked in natural language and the developed query and the returned rows narrated and described by a purely tabulated form. The disturbing omission usually comes from a low-code platform does not have such an embedding variable, Oracle APEX, where the data and application are to be merged from start to finish.

Therefore, the following literature review aims to cover, in brief, the three main streams on which the current work is based. In summary, the first is translating natural language into SQL because, given the broad range of researches over this domain, it fits best with this review. The second one is data-to-text generation, in which the expression is a persistent doubt over the fidelity of words to numbers. The third aspect, which is the most recent one, concerns the embedding of AI into the low-code development environment, APEX, particularly. Apart from all that, this literature review focuses on work relevant from 2020 onward, when the field drastically accelerated with the advent of larger language models. Older works are no longer representative of the ability of such systems.

## 2.2 Question Answering for NL2SQL

Generally, this is a translation problem: given a natural language query, one knows the schema, constructs and executes the SQL command, and gets the correct answer returned. When a well-trained model produces ill-formed SQL based on grammar, it draws attention from the earliest attempts, which limits production. (Scholak, Schucher, Bahdanau (2021)) constrained generation modeling possibilities only to allowable characters during decoding so that there would be no syntactic errors in the generated SQL without retraining the model.

The situation changed with the advent of LLMs. Instead of training a specific parser, a general model, a schema, and a few-shot interaction may be used to prompt the generation of SQL code. According to (Pourreza and Rafiei (2023)), prompting can be improved by decomposing the SQL generation task into logical steps in which the model can review drafts of what it produces. DAIL-SQL (Gao et al., 2024), which similarly couples a reflective investigation of the matching between the natural language proposed by the user-narration and the schema with the prompt selection mechanism based on such representations and the order of examples in the context to discover which types of prompts work. Hence, this study followed recent effective techniques without new architectural contributions.

This was taken further in BIRD (Li et al., 2023) to make benchmarking increasingly realistic: huge but imperfect database schemas thus force additional data; results are shown not only concerning their speed but also regarding the accuracy of the answers, immediately exposing some significant gaps between them and the best human experts. Judgment under realistic schema complexity was used in Spider 2.0 (Lei et al., 2025); only recently have such problems begun to be solved due to their near-professional relevance. Two of the most recent reviews

cover this shifting landscape along one further axis—from prompting models to fine-tuning: (Hong et al., 2024) and (Shi et al., 2024).

So, it is unclear what such an analysis would produce in practical business terms for the administrator. Most benchmarks are in English and domain-general; hence, those tables and terms will represent business entities and processes unknown to the models. The empirical results of BIRD and Spider 2.0 only further support this. Security is, in fact, the first blocker for applications: query generation, execution, and even prerun validation of queries for appropriate permissions are almost all but ignored in most literature. On the other hand, this search is too conceptual for Arabic and English.

### **2.3 Turning data into written reports: text generation**

The second capability works in the opposite direction. It turns structured data into readable text. Sharma et al. (2025) explain how text generation has developed from rule-based methods to statistical methods, neural models, and transformer-based models. They also used Google’s Gemini in a simple example to generate career suggestions and draft documents from users’ backgrounds. This example shows that transformer models can use information about the user to generate text. Today, these models are more widely used than older methods based on templates and manual features.

In data-to-text generation, the generated text must match the original data. Parikh et al. (2020) used the ToTTo dataset to study this. In ToTTo, some cells in a table are selected, and the model writes a sentence based on those cells. This helps check whether the generated sentence matches the information in the table. Nan et al. (2021) formulated the generation from sets of structured records into sentences in DART. Two surveys consolidate this concern. Li et al. (2022) review how

faithfulness is analyzed, measured, and improved; Ji et al. (2023) broadly map hallucinations across generation tasks, data-to-text among them, and mitigation strategies tried so far.

LLMs reopened the question of how best to pursue faithfulness. Zhao et al. (2023) studied table-to-text in realistic information-seeking scenarios and showed that LLMs can write statements more faithful than earlier fine-tuned systems and, moreover, that an LLM can act as a reasonable judge of faithfulness and provide step-by-step feedback on the draft of another model. Lin et al. (2024) reviewed neural data-to-text systems. They explained that these systems first decide what information to include and then decide how to write it. Roychowdhury et al. (2024), through ERATTA, used enterprise data retrieval with checks to reduce incorrect or made-up answers. This is relevant to this study because reliable reporting requires the generated text to match the available data.

Oyewole et al. (2024) reviewed the use of NLP in financial reporting. Their study covered tasks such as sentiment analysis, summarization, and regulatory reporting. They found that NLP can improve reporting efficiency and accuracy, but consistency is still a concern. They also mentioned a query-based summarization system that achieved about 46% ROUGE-1. Although their work is a review rather than a developed system, it is relevant because inaccurate information can cause serious problems in financial and administrative reports.

Yagamurthy, Azmeera, and Khanna (2023) looked at how structured organizational data can be turned into readable reports. Their study covered data such as financial records, business intelligence information, healthcare KPIs, and customer feedback. The process includes cleaning and organizing the data before generating the text. They also discussed challenges such as domain-specific language,

understanding the context, data quality, and ethical issues. These challenges are also important for administrative reporting. However, their study does not focus on Oracle APEX or bilingual reporting, which leaves room for the work proposed in this study.

There are still some gaps in reporting. One issue is that the generated text may not always match the actual data. This is important because one wrong number in an executive report can reduce users' trust in the system. Most studies focus on English and use general datasets such as Wikipedia. There is less research on administrative and financial reports, and very little research on bilingual reporting. Few studies also connect the generated summary directly to live data in a real application.

## **2.4 Bringing AI into Oracle APEX and low-code platforms**

The third area is the newest and least explored. Low-code platforms make it possible for people with limited programming experience to build applications using visual development tools. Oracle APEX is one of the leading low-code platforms because it runs directly on Oracle Database. In recent years, Oracle has added generative AI features to APEX, and a small number of studies have begun to explore their use.

Yeruva (2025) gives an overview of how AI can be used in Oracle APEX. The review discusses predictive analytics, sentiment analysis, recommendation systems, NLP-based interfaces, in-database machine learning, and the use of external models through REST APIs. However, the study does not focus on one specific system or test a complete solution. Instead, it gives a general view of AI applications in APEX and points out challenges such as integration, explainability, and scalability.

Keshireddy (2025c) used a large language model in the Oracle APEX development environment to help developers with coding. The system helps them create, understand, and improve code. The study reports faster development and fewer coding errors. However, like Yeruva's work, it focuses on application development rather than using natural language to query data or generate reports. Still, it does make a useful contribution by showing that LLM assistance inside APEX is not just at a proof-of-concept level but measurably accelerates real development work.

Keshireddy(2025d) presents an AI assistant created within Oracle APEX based on Retrieval-Augmented Generation (RAG). Instead of relying on a model's fixed training knowledge, this system, at query time, retrieves relevant content from organizational documents and records and uses it to generate context-aware and accurate answers. The claimed benefits are enhanced response accuracy, a better understanding of context, and improved user satisfaction. RAG is a methodology this thesis directly invokes, using Oracle 23ai's vector search to pull relevant history and schema documentation into each prompt; hence, this paper is direct evidence that such an integration can work within APEX and not just in stand-alone LLM settings.

Keshireddy (2025a) is the most comprehensive APEX paper, presenting a modular NLP architecture for APEX around intent recognition and context-aware responses with lightweight pipelines for intent detection, entity recognition, and adaptive dialogue. Four use cases are implemented: triggering predefined workflows from natural language, retrieving data on request, intelligently filling forms, and answering user queries with a conversational agent. It is also evaluated using metrics that any committee will recognize as real: task completion time, interaction error rate, intent-recognition precision, latency, and throughput. Although the focus

is on interaction and usability rather than report generation, this is the most helpful template for how to structure and measure an APEX-plus-NLP system in the set.

In another study, Keshireddy (2025b) goes from interaction to data integrity in the *Journal of Internet Services and Information Security (JISIS)*. Unlike the rigid PL/SQL rules most APEX systems use, this paper presented an adaptive AI framework within Oracle APEX to automate data transformation and validation based on semantic, statistical, and constraint-based techniques. It has been validated on 980,000 records in HRMS, CRM, and Accounting modules, with transformation accuracy exceeding 92% in HRMS and CRM and 94.6% in Accounting. Besides, the framework offers encryption, access control, and audit logging and adapts automatically to changes in the schema. This relates to two objectives of this thesis: the need to deal reliably with real enterprise data and ensure that the system's operations are secure and auditable. It is the most evidence-based of all the APEX papers and the most explicitly enterprise-scale.

Ashraf Syed (2025) is one of the studies most closely related to this thesis. The study focuses on the APEX AI Assistant introduced in APEX 24.1. It presents a proof-of-concept chatbot that converts natural language questions into executable SQL queries. The study reports 92% accuracy and a 50% reduction in report generation time based on an enterprise example. It also discusses challenges such as data privacy, model accuracy, and lack of technical skills, and suggests prompt engineering and Oracle's cloud infrastructure as possible solutions.

Although the topic is closely related to this thesis, the study is a single-author proof of concept, and the reported results are not supported by a clearly described evaluation method. Therefore, the reported numbers can be considered promising, but they should not yet be treated as confirmed results.

A broader lesson comes from beyond APEX. Rao et al. (2024) built LowCoder, combining a visual interface with a natural-language interface for assembling machine-learning pipelines and conducted a user study comparing the two. Their finding generalizes well beyond their setting: a natural-language interface enabled people to discover operators and options that would otherwise be hidden in menus, but they still depended on the visual layer to read and correct what the system produced. The implication is direct for a reporting tool: a natural-language front end will work only if the person can see and fix what the machine generated. Hence, the case for letting a user see the SQL before running it and compare the summary with the numbers.

The AI-in-APEX literature is small, recent, and mainly proof of concept. Most of it has been aimed at building applications or smoothing interaction rather than generating reports from data, and it is almost entirely in English, as in the other two areas.

## **2.5 Synthesis and research gap**

Each domain stands well by itself. NL2SQL research has developed mature approaches and increasingly realistic benchmarks. However, existing evaluations still focus predominantly on clean, English-language, and relatively generic datasets. Practical challenges, such as validating generated queries against complex and messy enterprise schemas and supporting Arabic-language queries, remain insufficiently explored. Similarly, data-to-text generation has advanced to the point where large language models can produce coherent and grounded summaries from structured data. Nevertheless, ensuring the faithfulness of generated narratives remains an open challenge, while administrative and bilingual reporting contexts have received limited research attention. AI integration within low-code platforms is also still at an early stage. The work most closely related to this thesis, an

SQL-generating chatbot implemented in Oracle APEX, remains primarily a proof-of-concept, lacks rigorous empirical evaluation, and does not address narrative report generation or bilingual Arabic-English interaction.

This thesis lies in the gap. To the best of the author's knowledge, no work has combined a validated natural language-to-SQL front end with a faithfulness-controlled data-to-text generation back end within a low-code Oracle APEX application for administrative and executive reporting in Arabic and English. The pieces exist in isolation and are usually tested in isolation from any working enterprise system. Hence, the present thesis contributes to building and evaluating the entire pipeline in its actual environment, which applied research backed by firsthand APEX experience is well positioned to provide.

## **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 the 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 that 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 of the cycle from start (user input) to end (report and data display).

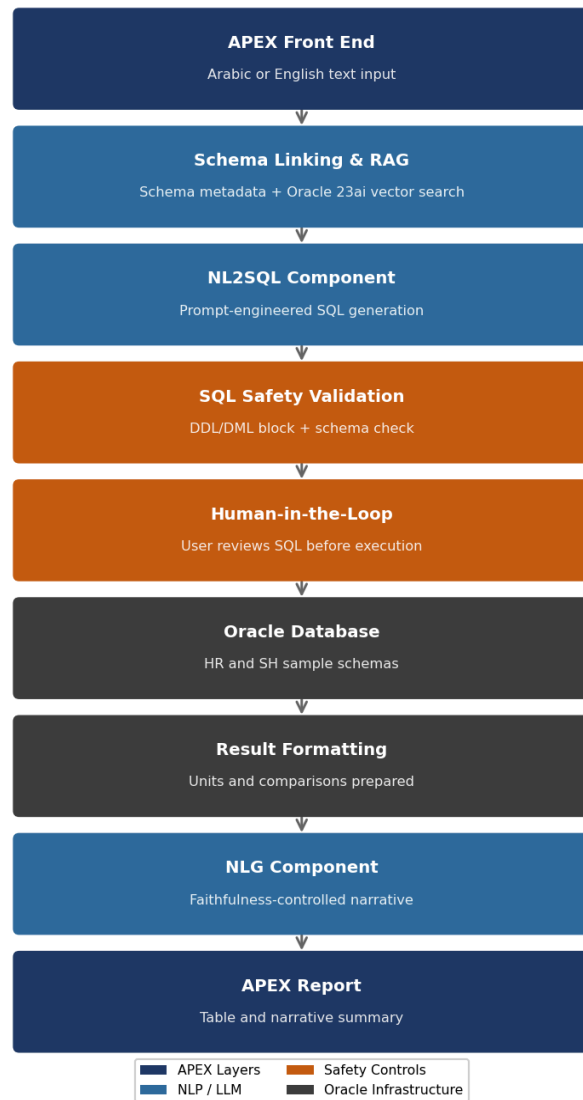


Figure 2.6.1 System pipeline architecture for the proposed NLP-based reporting framework

### 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**

Thereafter, the prompt goes to the large language model for step-by-step thinking: which tables to JOIN, how to calculate growth with conditional aggregation by year instead of with subqueries, how the promotion was LINKED with a LEFT JOIN since sales do not always have a promotion code, and how NULLIF was used in the growth percentage formula to avoid division by zero. The result is as follows:

*Table 2.6.1 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 is not a typical reporting query; it involves conditional aggregation and derivative columns with zero-division protection and per-row discounted calculations performed by aggregation. It does not involve DDL/DML keywords, dynamic SQL, or references to foreign tables or columns- only locally referenced ones. Human validation is needed.

#### **2.6.1.4 Human Confirmation**

The system then allows presenting the following explanation in layman's terms: "Compare the revenues for the years 2000 and 2001 for each product subcategory; calculate the percentage variation between the two years, and add the average

discount granted by the promotion in sales for each subcategory. It reads from 4 tables and does not change anything." The user allows them to make changes and accepts them. The system is then allowed to run. Testing and developing all queries that check whether all conditions prescribed by the user are met for the query is among the first priorities in this area.

#### **2.6.1.5 Execution and Result Formatting**

Processing and Formatting the Result: Five categories are thus returned after the query; each row shows the category name. This is then further processed and narrativized; hence, this kind of output should be projected into the results page, further characterizing it as a currency symbol and rounding percentages. See Table 2 for details.

*Table 2.6.2 Execution and Result Formatting*

| <i>Product Category</i>                     | <i>Revenue 2000 (USD)</i> | <i>Revenue 2001 (USD)</i> | <i>Growth (%)</i> | <i>Avg Promo Discount (%)</i> |
|---------------------------------------------|---------------------------|---------------------------|-------------------|-------------------------------|
| <b><i>Software/Other</i></b>                | \$9,124,200               | \$11,456,800              | +25.6             | 18.4                          |
| <b><i>Electronics</i></b>                   | \$8,345,600               | \$9,876,300               | +18.3             | 14.2                          |
| <b><i>Hardware</i></b>                      | \$6,782,100               | \$7,432,900               | +9.6              | 12.7                          |
| <b><i>Peripherals &amp; Accessories</i></b> | \$4,123,400               | \$4,398,200               | +6.7              | 15.9                          |
| <b><i>Photo</i></b>                         | \$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

Displaying the Report: The summary report developed with APEX will present the sales data and narrative in step 5 together with that in step 6 after completion. It will take her no more than seconds to learn that two categories grew fast, two grew slowly, and one shrank, thanks to the promotion discounts- all this with no SQL writing or ticket creation. If further drilling down to the category or region is desired, she simply asks the next question, and the loop starts again from step 1.

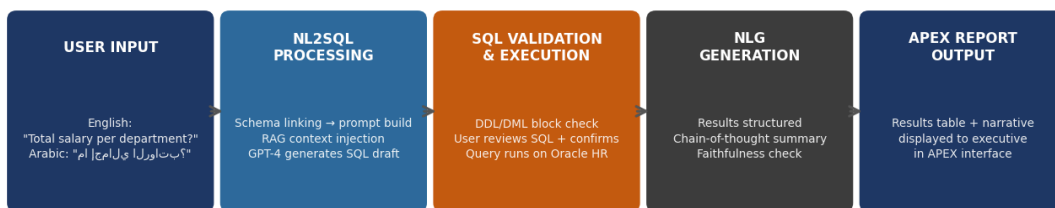


Figure 2.6.2 Complete cycle from user input to executive report for the worked example scenario

This is the case when several research questions relate to each other. As for RQ 1, in step 3, this would be the use of Oracle SQL statements with all JOINS, NULLIF, and conditional aggregation applied to the schema of Oracle in real life from a naturally posed question. Step 6 modestly refers to the table and narrates each sub-category one by one without any fiction, as per RQ2. Step 3 will consider the entire pipeline shown in Figure 1, showing the architecture for managing such complex queries safely as part of RQ3.

### *Appendix: Synthesis Matrix*

*Table 2.5.3 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

### [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](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>

