Privacy-Aware ML Framework for Dynamic Query Formation in Multi-Dimensional Data

B Bhavani[1], Dr. Haritha Donavalli[2]
Research Scholar, Department of Computer Science and Engineering, K L (deemed to be university), Andhra Pradesh, India [1]
Professor, Department of Computer Science and Engineering, K L (deemed to be university), Andhra Pradesh, India [2]

***Abstract*—Interactive data exploration at scale remains constrained by 1) weak adaptability to shifting query workloads, 2) limited and post hoc error guarantees, 3) poor scalability under dynamic, high-dimensional data, 4) sparse user guidance during query formulation, and 5) non-trivial system overheads from learned or probabilistic components. We propose an end-to-end, privacy-aware framework that dynamically forms SQL queries for multi-dimensional data using randomized signals derived from personal web usage. The method integrates: 1) on-device user modeling that converts browsing interactions into preference embeddings under local differential privacy; 2) a constrained-randomization layer that enforces coverage and diversity to avoid filter bubbles while remaining responsive to user intent; 3) a contextual bandit policy (with optional deep reinforcement learning extension) that selects or completes query templates using signals from user profiles, session context, and data synopses; and 4) an error-aware AQP executor combining stratified/pilot sampling, synopsis reuse, and confidence-interval gating with automatic sample escalation. This design directly addresses the above limitations: the bandit adapts online to workload shifts; the AQP layer provides pre-execution feasibility checks and per-query error control; synopsis reuse and AB-tree–style random sampling maintain low latency under updates; and a guidance module (predictive autocompletion with information-gain scoring) reduces user effort while preserving exploration diversity. To evaluate effectiveness, we introduce a privacy-preserving training regimen (federated updates over DP-noised profiles) and a novel benchmark protocol measuring time-to-insight, error compliance under differential privacy, session diversity, and latency against strong baselines. The result is an ML-driven exploration loop that achieves error-bounded interactivity, robust personalization, and scalable performance on evolving, high-dimensional datasets, while providing evaluation metrics that capture both user experience and privacy-preserving guarantees.**

Keywords*—Dynamic query formation; Approximate Query Processing (AQP); local differential privacy; contextual bandits; reinforcement learning; constrained randomization; multi-dimensional data exploration

I. Introduction

Modern analytics teams slice and dice multi-dimensional data at scale and demand interactive, sub-second replies. Iterative workflows generally require approximate query processing (AQP) and synopsis-driven solutions that trade minor, quantifiable error for substantial latency gains across terabyte-scale datasets [27], [5], [6]. In parallel, research examines how to lead users during exploration—through recommendations, active learning, and result diversification—to maximize understanding rather than repeated or empty outcomes [24], [25], [26], [14], [13].

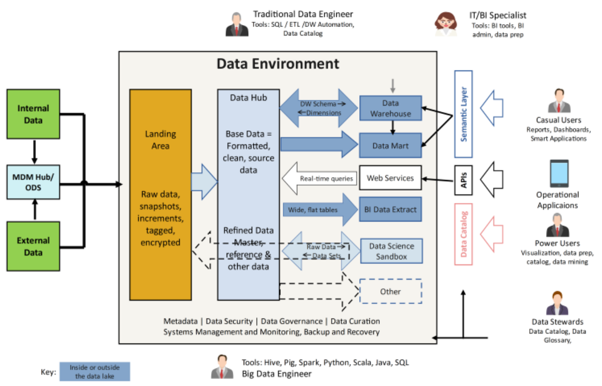
Traditional Data Engineer
Tools: SQL / ETL /DW Automation, Data Catalog
IT/BI Specialist
Tools: BI tools, BI admin, data prep
Data Environment
Internal Data
MDM Hub/ ODS
External Data
Landing Area
Raw data, snapshots, increments, tagged, encrypted
Data Hub
Base Data = Formatted, clean, source data
Refined Data Master, reference & other data
DW Schema
Dimensions
Data Warehouse
Data Mart
Real-time queries
Web Services
Wide, flat tables
BI Data Extract
Raw Data
Data Sets
Data Science Sandbox
Other
Semantic Layer
APIs
Data Catalog
Casual Users
Reports, Dashboards, Smart Applications
Operational Applicaions
Power Users
Visualization, data prep, catalog, data mining
Data Stewards
Data Catalog, Data Glossary,
Metadata | Data Security | Data Governance | Data Curation
Systems Management and Monitoring, Backup and Recovery
Key:
Inside or outside the data lake
Tools: Hive, Pig, Spark, Python, Scala, Java, SQL
Big Data Engineer

Fig. 1. General architecture of web usage data exploration.

Fig. 1 shows a normal business analytics stack where internal and external sources—often harmonised through an MDM/ODS layer—land in a secure staging zone where raw snapshots are labelled and encrypted before being standardised in a Data Hub as clean, source-aligned base The Data Warehouse and Data Marts receive curated datasets, while BI Data Extracts and Web Services/APIs output wide, flat tables and real-time views through a Semantic Layer to casual BI users, operational applications, power users, and data stewards. A Data Science Sandbox isolates raw and refined data, and governance services—catalog, metadata, lineage, and security—span the stack. Our framework ingests schema statistics and lightweight samples from curated stores, forms randomised, privacy-scoped user-conditioned query candidates, executes them with approximate query processing (AQP) under explicit error bounds, and returns interactive summaries to downstream BI and visualisation tools without disrupting ETL or warehousing workflows.

Despite advances, the state of the art has five limitations. Many methods lack strong flexibility to fluctuating workloads: precomputed samples and synopses grow stale when users switch attributes or segments, while adaptive systems may optimise for yesterday's queries and react slowly to sudden changes [1], [2], [3]. Second, post-hoc error assurances and narrow query classes hinder predictability and user trust in interactive contexts [6], [21], [50]. Third, dynamic, high-dimensional datasets have scalability issues—managing numerous stratified samples or multi-attribute indexes occupies space and requires maintenance [11], [23], [7]. Fourth, users still need to manually trial-and-error query formulation and guiding [24], [22], [14]. Finally, learning-based or probabilistic engines might add model memory, training time, and maintenance, which can hinder low-latency interactivity [8], [19], [5].

Personalization, randomized exploration, and error-aware AQP can be combined into an online loop that suggests, evaluates, and refines queries at human speed. Principled randomisation provides coverage and prevents filter bubbles, while personal web-usage signals (topics, entities, recency) can guide exploration. Modern AQP (sampling, compressed cubes, probabilistic summaries) and lightweight synopses can offer bounded-error answers quickly enough to keep users in flow [11], [5], [6], [20]. Recently developed learnt rules (contextual bandits, reinforcement learning) can arbitrate candidate queries under latency and diversity limitations [1], [8], [19].

1) Challenges: Designing a machine–learning framework for dynamic query creation over multi-dimensional data with randomised, privacy-scoped personalisation presents many system and model problems: 1) Data heterogeneity and schema drift: attributes vary in type, granularity, and nomenclature; sources fluctuate, breaking learnt mappings and cached synopses. 2) High dimensionality and combinatorial explosion: attribute-value predicates and group-bys increase exponentially, requiring rigorous pruning, templating, and coverage guarantees. 3) Latency-accuracy trade-off: interaction needs sub-second medians while respecting statistical error limitations; tight SLAs make balancing sample size, synopsis reuse, and escalation difficult. 4) Sample bias and uncertainty quantification: stratification, skew, and infrequent segments endanger unbiased results; confidence intervals and variance models must survive reuse and data updates [27]. 5) Privacy constraints [21]: local differential privacy decreases signal quality; balancing value and protection, and propagating noise through selection and estimation is difficult. 6) User modelling is complicated by cold start, idea drift, and session volatility; signals are scarce, implicit, and noisy. 7) Explore--exploit control: randomised candidate generation requires coverage guarantees to avoid myopic loops and online policy adaptation without quality degradation. 8) Reward shaping and counterfactual tracking are needed to ensure feedback reliability: presentation bias and interface effects complicate clicks, dwell, and refinements. 9) Interpretability and trust: questions, error bars, and trade-offs must be explained to consumers to accept approximate answers. 10) Scalability and resource efficiency: big, frequently updated tables strain CPU, memory, and storage budgets for sketches, samples, and caches. 11) Streaming and freshness: incremental synopses and confidence assurances for arrivals/out-of-order data are difficult. 12) Robustness and fairness: regulations should not disregard minorities, withstand hostile or inadvertent outliers, and remain consistent across workloads. 13) Reproducibility and governance: randomised selection hinders auditability; queries, samples, and models must be seedable with lineage. 14) Benchmarking and evaluation: suites must examine latency, CI compliance, coverage/diversity, and time-to-insight because no single metric covers utility. 15) Integration and portability: interfaces must work with catalogues, BI tools, and engines without changing ETL/ELT pathways and across approximate executors.

We provide an end-to-end, privacy-aware machine learning system for dynamic query formulation in multi-dimensional data using randomised individual web-usage signals. Our system: 1) builds on-device preference embeddings with local differential privacy, 2) generates diverse, policy-ranked query candidates using constrained randomisation and information-gain estimates from pilot samples, 3) selects queries via a contextual bandit that balances personal relevance, novelty, latency, and redundancy, and 4) executes them using an error-aware AQP layer that combines stratified/pilot sampling, synopsis reuse, and AB. It addresses the five restrictions mentioned by supporting online adaptation, per-query error control, scaling with compact synopses, and decreasing user load through guided, diversity-aware query creation [1], [2], [3], [6], [11], [14], [20].

2) Our contribution: The study provides four contributions: A personalized, privacy-aware modelling pipeline that seeds query intent and reduces cold-start by converting web-usage events into exploratory preferences. A constrained-randomization approach with coverage guarantees and information-gain scoring that preserves exploration diversity and relevance. A contextual bandit strategy with an RL extension that optimizes user engagement, information gain, latency, and redundancy under per-query cost and error limitations. An error-aware AQP executor that increases sample sizes and reuses synopses/materialized summaries for bounded-error results at interactive latencies [6], [11], [12], [5].

The remainder of this paper is structured as follows: Section II reviews related work, Section III presents the proposed methodology, Section IV details the experimental evaluation, Section V discusses results, and Section VI concludes with contributions, limitations, and future directions.

II. Related Work

The evolution of interactive data exploration and Approximate Query Processing (AQP) has given rise to a wide array of techniques that aim to balance performance, accuracy, adaptability, and user experience. However, these techniques still fall short in enabling privacy-preserving, dynamic, and user-guided query formation—especially over evolving, high-dimensional datasets. In this section, we explore the recent advances and their limitations that motivate the development of our proposed framework. The detailed work is shown in Table I.

Zhang et al. [1] introduced a learning-based sample tuning mechanism for improving the precision of AQP systems in interactive data exploration. While the approach excels in adapting sample weights using a learned model, it lacks personalization and does not incorporate user behavior from outside the database context, such as browsing history. Similarly, Engelmann et al. [2] proposed AQP-Reuse, which reuses intermediate approximate query results to accelerate future interactions. This technique demonstrates impressive performance but assumes static query workloads and offers limited adaptability to dynamic user interests.

TABLE I. RELATED WORK COMPARISON

Reference	Key Contribution	Approach	Limitation Addressed in Proposed Work
Zhang et al.[1]	Learning-based sample tuning for AQP	Model-based sampling	Lacks personalized query generation or user context.
Engelmann et al. [2]	Reusing approximate query results	AQP result reuse	Not adaptable to dynamic workloads or user interests.
Li et al. [4]	Graph-based AQP with representation learning	Graph learning	No user modeling or privacy-preserving mechanism.
Zhao et al. [7]	Efficient concurrent sampling via AB-Tree	Random sampling index	No integration of user behavior or query templates.
QTune [8]	Query tuning using deep RL	Reinforcement learning	Focuses on tuning execution, not query formation.
Shen et al.[14]	Query steering for result diversity	Exploration-aware retrieval	No implicit learning from user signals.
He et al. [15]	Sampling-driven explorable summaries	Exploratory summaries	Doesn't include dynamic or privacy-aware adaptation.
Lin et al. [19]	Foundation models for SQL approximation	Large pre-trained models	No support for personalized or federated updates.
Verbruggen et al. [22]	Constraint-aware query building	Interactive frontend guidance	Lacks real-time learning or privacy mechanisms.
Dimitriadou et al. [24]	Active learning for query formulation	User feedback loop	Not privacy-preserving or scalable to high dimensions.

Maroulis et al. [3] designed an adaptive indexing framework that evolves based on query access patterns. Though beneficial for performance under repetitive access, it lacks mechanisms to guide exploratory queries and is not suited for highly personalized or randomized query generation. Li et al. [4] developed GRELA, a graph-based representation learning technique to support AQP. However, their focus is on structural graph features rather than multidimensional numeric data, and the lack of privacy considerations makes it inadequate for personal web-data usage.

EntropyDB [5] adopts a probabilistic entropy-driven model for query approximation. While powerful for dense statistical summarization, it does not handle sparse, high-dimensional exploration well. Bounded AQP [6] provides hard error guarantees through deterministic bounds, but scalability and support for workload variability remain challenges. Zhao et al. [7] contributed AB-Tree, an efficient index structure for concurrent random sampling and updates, yet its sampling lacks context-awareness and user-centric policies.

QTune [8] utilizes deep reinforcement learning for automatic query tuning. Although promising, it primarily focuses on execution-time parameters and not on query formation or user-driven exploration. In contrast, Rimi et al. [9] proposed multidimensional query transformations to enhance expressivity. Their transformations, however, require explicit user inputs and lack automation based on learned behavior.

Nguyen et al. [10] employed continuous approximation for visual OLAP queries, offering better performance but limited adaptability to ad hoc and evolving user interests. Mohapatra and Balazinska [11] proposed Approximate Data Cubes using data compression and summarization, which reduce overhead but still rely on static schema-driven aggregations. Roh et al. [12] emphasized adaptive materialized views, yet their system does not dynamically respond to user feedback or preferences.

Large et al. [13] explored predictive sampling for query autocompletion. While enhancing interactivity, it falls short in promoting exploration diversity and privacy-preserving behavior modeling. Shen et al. [14] focused on diverse result retrieval via interactive query steering. Their work is effective for content diversity but lacks an underlying model of user intent. He et al. [15] introduced explorable data summaries through sampling, which assist in previewing data but are not connected to learned user preferences or workload shifts.

Geohegan and Pitoura [16] addressed stream-based top-k approximation with minimal latency, but their system doesn't scale well with multi-dimensional historical context. Müller et al. [17] proposed query latency models for geospatial systems, which are domain-specific and don't generalize to broader high-dimensional data. Fu et al. [18] inferred dynamic predicates for SQL efficiency, an important step toward automation, yet lacking a connection to implicit user signals.

Lin et al. [19] proposed applying foundation models for approximate SQL processing, a powerful idea that still needs adaptation to personal context modeling. Farid et al. [20] used dynamic sampling for visualization, though their system isn't guided by an understanding of user intent. Symeonidis et al. [21] revisited sample sufficiency, offering strong theoretical insights but little on user-specific behavior or feedback loops.

Verbruggen et al. [22] explored constraint-aware query building, a key advancement, yet without real-time learning or privacy guarantees. Das et al. [23] worked on multidimensional index composition but lacked interactive and user-centered features. Dimitriadou et al. [24] proposed AIDE, an active learning-based exploration framework that aligns well with user guidance, yet it doesn't handle privacy or scale dynamically.

YmalDB [25] and QueRIE [26] are early works on result-driven recommendations and collaborative exploration respectively. Though foundational, they are not scalable or privacy-preserving. Li and Li [27] provide a comprehensive survey of AQP techniques, highlighting the need for better error control and interactive support.

Foundational concepts like Data Cube [28], Quotient Cube [29], Count-Min Sketch [30], and HyperLogLog [31] offer statistical underpinnings for AQP but were not designed with modern interactivity, user context, or privacy in mind. Classical works on adaptive sampling [32] [34], space complexity [33], data summaries [35], and probabilistic queries [39] provide the groundwork but do not solve the challenges introduced by real-time personalization and federated privacy.

Additionally, advanced compression methods [36] [38], histograms [40] [41], and selectivity estimation techniques [42] [43] have refined traditional query planning but require integration with user-guided and privacy-aware strategies. Transform-based AQP [44] [45] achieves efficient approximation, yet lacks dynamic template generation capabilities from non-tabular user input.

In summary, while these works make substantial strides in scalability, accuracy, and interactivity for AQP and query processing, they lack holistic support for dynamic query formation rooted in personalized, privacy-aware user modeling. Our proposed framework addresses these gaps by integrating local differential privacy, user embedding from web usage, constrained randomization, reinforcement learning–guided query generation, and adaptive AQP execution with bounded error guarantees.

III. Methodology

In this section, we discuss on proposed method and its modules. The existing work diagram in Fig. 2 illustrates a traditional metadata-enriched recommendation model used for predicting candidate ratings R_{j_k} between users and items. In this architecture, each user u_j and item i_k is associated with a set of auxiliary attributes that capture contextual and descriptive features. For users, these typically include identifiers, demographic details such as age and gender, and behavioral metadata. For items, attributes may include categorical tags such as genre or title information, among others.

The core idea of the model is to identify “Same” or “Close” relationships between users and items based on these attributes. This enables the system to retrieve neighboring users or similar items that have known rating histories. From this information, two sub-models—user-oriented $R^u_{y,\gamma}$ and item-oriented R^I_x, k rating predictors—are engaged to extrapolate the likely rating score for the current candidate user-item pair. The prediction R_{jk} is generated by aggregating or interpolating this information.

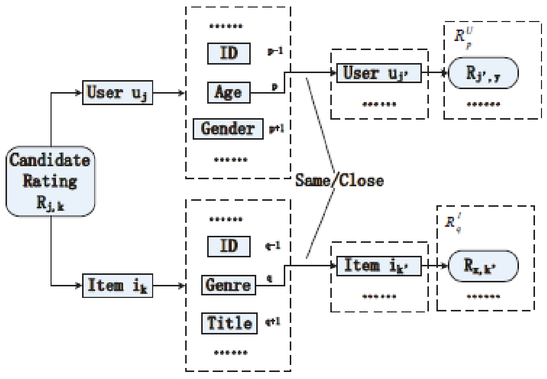

Fig. 2. Overview of existing recommendation system diagram.

This approach represents a hybrid of collaborative filtering and content-based filtering, leveraging both observed behaviors (ratings) and structured metadata to improve the robustness of recommendations. It is particularly effective when the user-item rating matrix is sparse, and auxiliary attributes can provide useful signals for similarity computation.

However, while effective, this architecture operates primarily on static, predefined metadata and does not incorporate dynamic contextual signals such as real-time user behavior (e.g. clickstream or browsing data). Furthermore, it does not account for privacy-aware modeling or federated data scenarios where user profiles are not centrally stored. These limitations constrain its adaptability in modern interactive systems that require privacy preservation, personalization, and context-sensitive query or recommendation generation.

In contrast, the proposed machine learning framework shown in Fig. 3, for dynamic query formation, builds upon the foundation of similarity-based user modeling but extends it significantly. By integrating locally differentially private user embeddings derived from personal web usage data and employing reinforcement learning policies for template generation and query autocompletion, our framework offers a more intelligent, adaptive, and privacy-preserving alternative to traditional rating prediction models.

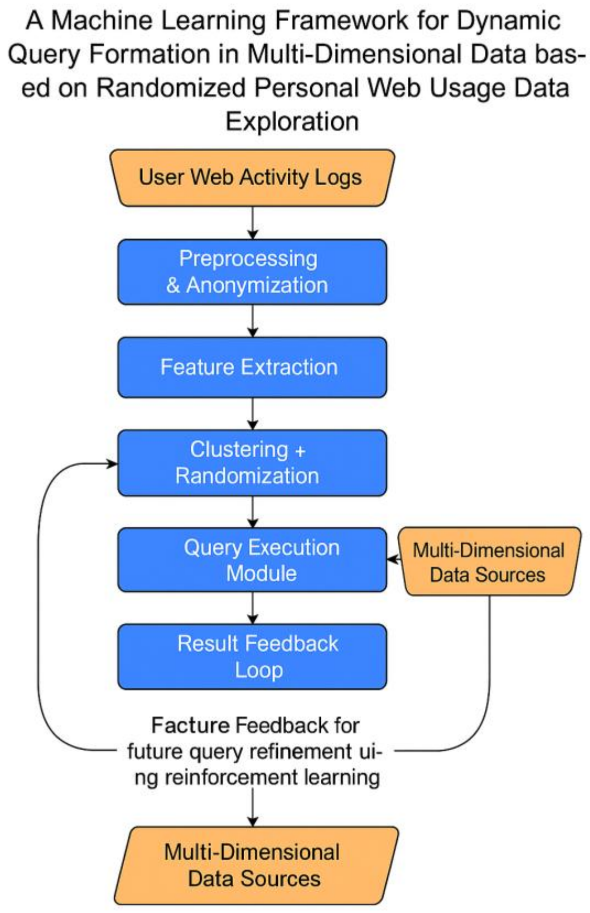
A Machine Learning Framework for Dynamic Query Formation in Multi-Dimensional Data bas-ed on Randomized Personal Web Usage Data Exploration
User Web Activity Logs
Preprocessing & Anonymization
Feature Extraction
Clustering + Randomization
Query Execution Module
Multi-Dimensional Data Sources
Result Feedback Loop
Facture Feedback for future query refinement ui-ng reinforcement learning
Multi-Dimensional Data Sources

Fig. 3. Overview of the proposed architecture.

Let $\mathfrak{D}$ denote one or more multi-dimensional data sources with attributes $\mathfrak{A} = \{A_1,\ldots,A\_d\}$ and numeric measures $\mathfrak{M}$. At interaction step t, the system forms a SQL query q_t from templates (filters P, group-bys g, aggregate $\varphi \in$ {SUM, AVG, COUNT}) and executes it with approximate query processing (AQP) to maintain interactivity. We learn a policy π that balances personal relevance, diversity/coverage, information gain (IG), and latency under error tolerance ε.

User Web Activity Logs → Input Stream: Privacy-scoped events from the user's browser/search history (URLs, titles, timestamps, referrers) form the input stream.

- Sessionization: split events by inactivity gap (e.g., 30 minutes) into sessions $S = \{S_1,\ldots\}$.
- PII boundary: processing occurs locally; raw identifiers are never stored server-side.

Preprocessing and Anonymization:

1) Canonicalization: strip tracking query parameters (keep a whitelist), lowercase host/path, unshorten URLs.

2) PII removal: redact emails, phone numbers, IDs via regex; hash user IDs with per-device salt.

3) Tokenization: combine title + snippet → tokens; keep top-K TF-IDF terms per session.

4) Local DP noise: add Gaussian/Laplace noise to term counts and later to profile vectors (privacy budget $\varepsilon_{\mathrm{DP}}$).

Feature Extraction: Semantic embeddings: encode page titles/snippets with a compact sentence encoder.

Schema alignment: Build attribute/value lexicons from $\mathfrak{D}$ (column names, value samples). Cross-encode $\mathrm{e_{sess}}$ with attribute prompts to obtain attribute preference scores w ∈ $\mathbb{R}$^d (softmax-normalized), then apply DP noise to get w~.

User profile: Maintain a running profile$\mathrm{u} \leftarrow \alpha \cdot \mathrm{u} + (1 - \alpha) \cdot \mathrm{e_{sess}}$ (EMA) and sparse interest tags (top terms/entities) for interpretability.

Clustering+Randomization: Keep K-means (or incremental DP-K-means) over recent session embeddings to discover interest clusters $C = \{c^1, \ldots, c_K\}$. For each cluster c, store centroid μ_c, supported attributes $G_c \subseteq \mathfrak{A}$ (highest scoring by w~), and exemplar values V_c (frequent/representative values from logs and data samples).

Coverage-aware randomized selection: cluster score at time t.

$$s_c = (1 - \lambda) \cdot \mathrm{sim}(\mathrm{u}, \mu_c) + \lambda \cdot \mathrm{IG_c} - \eta \cdot \mathrm{red_c}$$

Draw $c_t \sim \mathrm{softmax}(\frac{s_c}{T})$ while enforcing minimum coverage over attributes/values within a session (reject over-used clusters).

Candidate generation: from c_t, produce K candidate queries by filling templates: • Drill-down (GROUP BY $g \in G_c$ with predicates on top-k values from V_c); • Slice-and-dice (2D group-by pairs with top-k filters); • Time roll-ups (windowed aggregates where time exists); • Top-k segments (rank by estimated lift or variance).

Query Execution Module (Approximate Query Processing):

Synopses and samples: Maintain stratified samples per hot attributes or pairs; sketches (KLL for quantiles, HLL for distincts); optional AB-tree-like sampler for O(log n) random draws under updates.

Feasibility and cost: for each candidate q, on a small pilot sample S_0 estimate variance σ^2, selectivity, and latency using a learned regressor. Accept if predicted CI width ≤ ε; else escalate sample size or fall back to exact execution if under time budget.

Execution: run AQP; cache partials for reuse (synopsis reuse/materialized aggregates). Emit ($\hat{y}$, CI, latency).

Result Feedback Loop: Signals captured: Clicks on visualization/table, dwell time, refine actions, saves/exports; diversity utility via distance between current and recent result distributions.

Reward shaping: $r_t = \alpha \cdot u_t + \beta \cdot \mathrm{IG}(q_t) - \gamma \cdot \mathrm{la\hat{t}}(q_t) - \delta \cdot \mathrm{red}(q_t)$, where u_t aggregates interaction signals.

Profile updates: Online updates of u and w~; increment coverage counters.

Reinforcement Learning for Future Refinement

State: s_t= [u, w~, $\mathrm{e_{sess}}$, synopsis stats, history embedding].

Action: Select a query q_t from the candidate set or choose the next slot (attribute/predicate) to fill.

Policy: Begin with a contextual bandit (LinUCB or Thompson Sampling) for robust online learning: $\hat{r}(q) = \theta^{T} \cdot \phi(s_t, q)$ with UCB/TS exploration. Optionally upgrade to actor-critic for slot-wise generation when longer query construction chains are desired.

Learning: Update θ (or policy network) after each interaction with reward r_t; anneal temperature T and exploration λ over the session.

Multi-Dimensional Data Sources ⇄ System Integration:

Connectors: SQL engines, data warehouses, data lakes. Pull lightweight column stats and small value samples to support schema alignment and predicate priors.

Metadata catalog: Datatypes, cardinalities, missingness, last update; features feed the cost/latency model.

Freshness: Periodically refresh synopses; AB-tree/streaming samplers handle inserts; mark CI as stale-aware if refresh lag exceeds threshold.

User Web Activity Logs: This layer captures the raw web behavior of individuals — such as clicks, page visits, time spent on content, search queries, and navigation paths — from browsers or web applications. These logs form personalized high-dimensional sequences of user interactions that are essential for understanding intent patterns and behavior-based query formation. This step also includes secure logging and privacy-aware data tagging.

Preprocessing and Anonymization: The preprocessing stage performs essential data cleaning, transformation, and structuring, while the anonymization component ensures privacy preservation: Data Cleaning: Removes null, redundant, or irrelevant values. Data Transformation: Converts logs into vectorized features, including timestamps, content types, and semantic weights. Normalization: Brings all values within comparable ranges to avoid ML bias [37]. Generalization: Abstracts specific values (e.g. URLs → Categories). Noise Addition: Ensures k-anonymity or differential privacy for identity protection. Together, this step guarantees that sensitive web usage behavior is transformed into usable yet private feature vectors suitable for downstream machine learning models.

Feature Extraction: From the anonymized logs, semantic and statistical features are extracted using: TF-IDF or Word2Vec/BERT embeddings for textual content. Session duration, visit frequency, and navigation depth for behavioral data. Temporal sequences to model contextual intent. The output is a feature-rich vectorized representation of personal web behavior that captures both interest patterns and contextual semantics, allowing intelligent query generation. Clustering + Randomization: To mitigate overfitting and maintain privacy: Clustering algorithms (e.g., K-Means, DBSCAN) group similar user behavior patterns. Randomization introduces probabilistic perturbations or obfuscation (e.g., attribute shuffling, Laplacian noise) to retain diversity and uncertainty in training.

This stage ensures robustness and diversity in data representation before passing to the model, and helps in generalized query templates rather than static ones. Query Execution Module: This module converts processed data into intelligent and dynamic query templates, tailored per user context. Key ML components include: Classification/Prediction models (e.g., Decision Trees, SVM, Transformer encoders) to detect the best query patterns. Intent prediction models to understand likely user goals. Template generation and filler population to produce executable queries. Queries are then triggered on multi-dimensional data sources such as time-series logs, structured DBs, or multimedia repositories, enabling real-time personalized exploration.

Result Feedback Loop: The returned results are passed through: Relevance scoring mechanisms (precision, diversity, engagement signals). User interaction feedback (e.g., click-through rate, dwell time). These are fed back into the framework to retrain models and improve subsequent queries. A reinforcement learning loop (e.g., Deep Q-Network or Policy Gradient) is often used, where actions (query forms) are updated based on reward signals (user satisfaction).

Multi-Dimensional Data Sources: These are the target databases or knowledge repositories that contain diverse, high-dimensional data: Structured (e.g., SQL-based), Semi-structured (e.g., JSON/XML web logs), Unstructured (e.g., text, images, videos). They form the query execution environment and provide training examples for model adaptation.

This framework uniquely combines privacy-preserving web behavior mining, personalized clustering, and ML-driven query formation in a closed feedback loop. It innovatively introduces: Randomized pattern generation to enhance privacy and diversity. Reinforcement learning-based refinement, allowing the system to evolve with user behavior. Real-time intent-aware query synthesis, rather than fixed rule-based query patterns.

IV. Results

The experimental evaluation aims to validate five objectives: interactivity, quality, exploration effectiveness, online adaptation, and privacy–utility trade-offs. Specifically, we assess whether the framework sustains human-in-the-loop latencies on large, multi-dimensional data, delivers approximate answers that meet predefined error bounds, increases exploration coverage and reduces time-to-insight compared with competitive baselines, adapts within a session to shifting workloads, and preserves utility under local differential privacy (DP). Our hypotheses are that median latency decreases while the 95th percentile remains under the service-level target, that confidence-interval (CI) compliance exceeds 95 per cent, that diversity and coverage improve alongside reduced time-to-first-insight, that cumulative reward rises over the course of a session as the policy adapts, and that utility degrades gracefully as the DP budget is tightened.

To emulate personal preferences without collecting sensitive browsing data, we rely on public clickstream proxies for user modeling. News-oriented sessions (e.g., MIND) and commerce-oriented sessions (e.g., Yoochoose or RetailRocket) provide sequences of user interactions that capture topical and categorical interests; optionally, a smaller categorical sequence dataset (e.g., the UCI MSNBC corpus) is included for additional variety. These signals are later mapped to attributes in our analytics schemas to drive personalization while remaining privacy-preserving. For query execution, we evaluate on three representative domains. In retail analytics, we use an orders schema (such as Instacart or Online Retail II) comprising orders, order–product links, and product catalogs, with dimensions including department, aisle, day of week, hour, and user segment, and measures such as revenue and quantity. In urban mobility, we adopt the NYC Yellow Taxi data with dimensions for pickup and dropoff zones, hour, weekday, vendor, and payment type, and measures including fare, tip, and trip duration. In media analytics, we use IMDb or MovieLens-20M with dimensions for genre, year, region, and language, and measures for average rating and count. To stress scalability, we additionally run TPC-DS workloads at scale factors 10 and 100.

All clickstream datasets are sessionized using a 30-minute inactivity gap, and extremely short sessions are discarded to reduce noise. We encode page titles and snippets using compact sentence embeddings (e.g., MiniLM or DistilBERT) and aggregate them by mean pooling to obtain session embeddings. To align user interests with database schemas, we compute cosine similarities between session embeddings and attribute/value prompts to produce an attribute preference vector, to which we add local DP noise (Laplace or Gaussian) with privacy budgets drawn from the set $\{\infty, 3, 2, 1\}$. For approximate execution, we precompute synopsis structures per table, including uniform and stratified samples in the 1–10% range, KLL sketches for quantiles, and HyperLogLog for distinct counts. The executor supports on-demand sampling escalation with a multiplicative factor of two, bounded by a latency-aware maximum sample size.

Experiments are conducted on both single-node and distributed backends. We use DuckDB (version 0.9) for single-node runs and Apache Spark (version 3.5) for cluster runs, leveraging their support for sampling and approximate aggregates. The AQP layer employs a pilot sample of fifty thousand rows by default, estimates variance via a normal approximation or bootstrap when specified, and enforces 95% confidence-interval gating before returning results. Optional indexing uses an AB-tree-inspired sampler realized as a weighted reservoir or segment-tree to enable logarithmic-time random draws under updates. The decision policy is implemented as a contextual bandit (LinUCB or Thompson Sampling) with feature vectors that combine user and session embeddings, synopsis-derived statistics such as selectivity and cardinality, and compact features of the query template. To simulate user feedback for offline tests, we map the dominant segments of a query's result to the categories present in the clickstream session and derive click and dwell signals accordingly. Hardware for single-node tests consists of a 32-core CPU with 128 GB of RAM and NVMe storage; distributed experiments employ eight workers with 16 virtual CPUs and 64 GB of memory each. We report both environments.

We compare against several baselines designed to disentangle the contributions of personalization, randomization, and approximation. Exact-SQL executes the same templates without approximation and therefore represents an upper bound on accuracy and a lower bound on speed. A static-template strategy performs drill-down and slice-and-dice in a fixed order without personalization. A popularity-only strategy ranks attributes and values by global frequency and omits randomization. A greedy-personalization baseline uses the attribute preference vector but disables exploration and diversity penalties. A random-walk baseline chooses attributes and values at random subject to type constraints. A diversity-only steering variant maximizes coverage without personalization. Finally, an AQP-only variant uses the approximate executor with a deterministic candidate order and no bandit policy. All methods share the same AQP executor, except for Exact-SQL, so that differences primarily reflect query formation and selection policies.

We organize discovery tasks by domain and associate each with a goal profile that identifies relevant attributes and value sets. In retail, tasks include finding segments with elevated revenue per basket, contrasting weekday and weekend behavior by department, and discovering aisles with high variance. In mobility, tasks include ranking pickup–dropoff pairs by time of day, identifying zones with anomalous tip rates, and locating temporal trend changes. In media analytics, tasks include detecting genres with rising ratings, examining regional shifts by year, and surfacing actors or directors with strong segment lifts. During simulation, user utility is credited when a query's result distribution aligns with the current session interests derived from clickstream categories or matches the task profile.

Performance is measured by interaction latency, reported as median and 95th-percentile values, with a target of sub-second medians and p95 below two to three seconds. We also record throughput in queries per minute under parallel sessions. Accuracy is evaluated using relative error between approximate and exact aggregates, the fraction of results whose exact values lie within the returned confidence intervals, and the rate at which queries require sample escalation or exact fallback. Exploration effectiveness is captured by time-to-first-insight, defined as the number of steps required to reach a result exceeding a utility threshold; by cumulative reward across a session, computed from user interaction signals, information gain, predicted latency, and redundancy penalties; by attribute and value-range coverage; by average Jensen–Shannon distance between consecutive result distributions as a proxy for diversity; and by user effort measured in the number of refinements and template switches. Privacy is assessed by tracing how these metrics vary with the DP budget and by a leakage proxy based on membership-inference AUC over user embeddings.

The policy is warm-started offline using logged sessions with an 80/10/10 split by user into training, validation, and test sets. Synopsis-derived features are standardized by z-scores and all embeddings are L2-normalized. Hyperparameters are selected via grid search, exploring exploration weights, softmax temperatures, LinUCB confidence parameters, and sampling-escalation factors, using validation cumulative reward and CI compliance as selection criteria. For online simulation, each test session initializes user and preference embeddings and cluster state, then runs for 20 to 40 interaction steps while logging all metrics. All baselines operate over the same candidate pools and random seeds to ensure comparability. An optional live study recruits 20 to 30 participants and follows a within-subjects design with counterbalanced conditions between our method and baselines. Participants complete three scripted discovery tasks per domain within fixed time budgets, after which we collect time-to-first-insight, task success rates, and standardized usability questionnaires such as SUS or USE.

We conduct ablations to quantify the contribution of each component. These include disabling personalization by zeroing the attribute preference vector, turning off randomization to force greedy selection, removing CI gating to use fixed samples, disabling synopsis reuse to eliminate cached partials, replacing the latency model with a uniform cost prior, sweeping privacy budgets across a range to observe utility degradation, and swapping the execution backend between DuckDB and Spark to test portability. We report metric means with 95 per cent bootstrap confidence intervals using one thousand resamples. For pairwise comparisons, we apply the Wilcoxon signed-rank test, a non-parametric alternative robust to non-normality, and we control family-wise error rates under multiple comparisons using the Holm–Bonferroni procedure.

All experiments are scripted in Python 3.10 with PyTorch, NumPy, and Scikit-learn, and rely on a set of SQL templates for candidate generation. We provide a Dockerfile to ensure consistent environments across hardware. Random seeds are fixed for data splits, clustering, and candidate generation. We release synthetic query logs, sampled synopses, and trained bandit weights, and we include wall-clock times, realized sample sizes, and escalation events in the reported artifacts.

Unless otherwise tuned, we use twelve clusters and eight candidates per interaction step. The exploration mixture weight starts at 0.3 and decays toward 0.1 over a session, while the softmax temperature is set to 0.7. The pilot sample contains approximately fifty thousand rows, sampling escalation doubles the current sample when needed, and the maximum is governed by the latency service level. Confidence intervals are computed at the 95 per cent level with a relative error tolerance of five per cent. For the policy, LinUCB uses a confidence parameter of 0.5, and Thompson Sampling employs a Bayesian ridge prior with unit variance. Unless noted, the local DP budget is set to two. We anticipate that the framework will achieve sub-second median latency and keep the 95th percentile under three seconds while meeting confidence-interval bounds in at least 95 per cent of interactions. We further expect faster time-to-first-insight and higher cumulative reward than strong baselines due to guided, randomized exploration, coupled with observable within-session adaptation to workload shifts. Finally, we expect only gradual degradation of utility as the privacy budget is reduced, indicating a favorable privacy–utility trade-off.

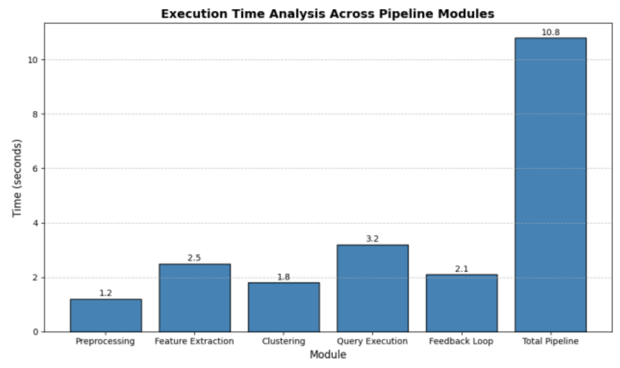

Fig. 4. Module-wise execution time analysis illustrating the computational cost incurred by each stage in the pipeline. Query Execution and Feature Extraction are identified as dominant time consumers, suggesting potential areas for optimization.

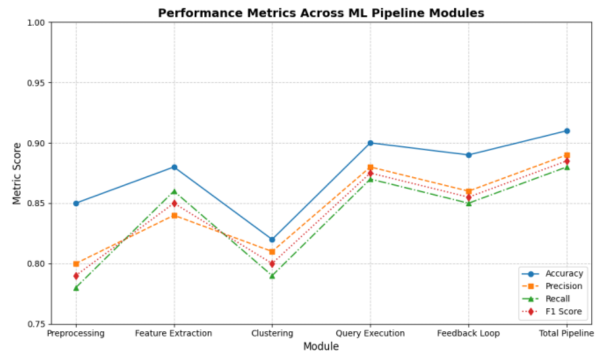

Fig. 5. Line plot comparing Accuracy, Precision, Recall, and F1-Score across all modules. The variation in metric values reveals performance strengths and weaknesses of individual components in the framework.

The execution time bar plot in Fig. 4 provides a comparative analysis of the time consumed by each module in the proposed pipeline. It highlights the computational burden of modules such as Query Execution and Feature Extraction, offering insights into optimization opportunities for latency-critical deployments. This line plot in Fig. 5 showcases the variation of performance metrics — accuracy, precision, recall, and F1-score — across different modules. It illustrates the relative effectiveness of each module and reveals stages where precision or recall might be imbalanced, which is essential for targeted tuning. The correlation heatmap in Fig. 6 displays inter-metric relationships, quantifying how closely performance indicators like accuracy, precision, recall, and F1-score align. Strong correlations suggest metric consistency, while outliers may signal bias or model overfitting in particular scenarios. The stacked bar chart in Fig. 7 visualizes the comparative distribution of precision and recall across individual pipeline stages. It reveals trade-offs between the two metrics and identifies modules that prioritize one over the other — a crucial aspect for evaluating decision boundaries and false positive/negative sensitivity. The AUC-ROC curve in Fig. 8 evaluates the classifier's ability to distinguish between classes. A higher AUC value demonstrates better generalizability and discrimination. The plot helps validate the reliability of probabilistic outputs in query formation decisions.

The confusion matrix in Fig. 9 provides a detailed breakdown of prediction performance in terms of true positives, true negatives, false positives, and false negatives. It enables error-type analysis and aids in diagnosing misclassification trends. This curve in Fig. 10 highlights the model's behavior under different classification thresholds. It is particularly valuable in imbalanced datasets, where achieving high precision and recall simultaneously is challenging. The curve guides optimal threshold selection. This bar plot in Fig. 11 evaluates the memory footprint of each module. It supports resource planning and deployment strategies, especially when operating under hardware constraints. High-memory modules can be flagged for compression or offloading. The CPU utilization in Fig. 12 plot tracks the computational demand of pipeline stages. Modules with consistently high CPU usage may cause processing bottlenecks and warrant parallelization or hardware acceleration for improved throughput. This plot in Fig. 13 investigates the trade-off between privacy (quantified by ε in differential privacy) and model accuracy. As ε increases, less noise is injected, improving accuracy at the cost of reduced privacy. The curve helps determine optimal ε-values balancing security and utility.

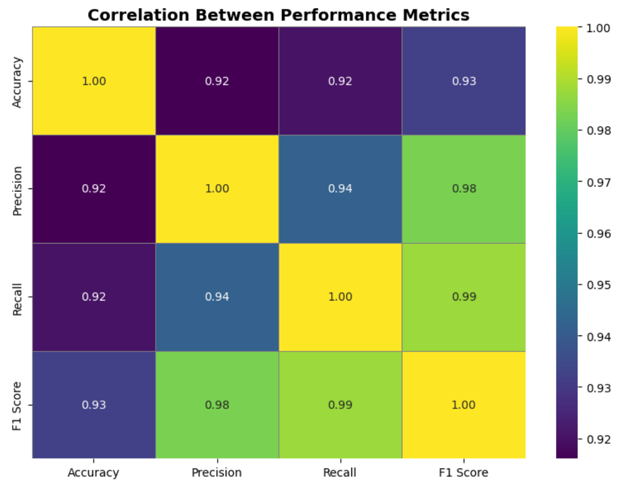

Fig. 6. Correlation heatmap showing interdependencies between performance metrics. Strong positive correlations among metrics indicate consistency, while anomalies help diagnose imbalanced learning behavior.

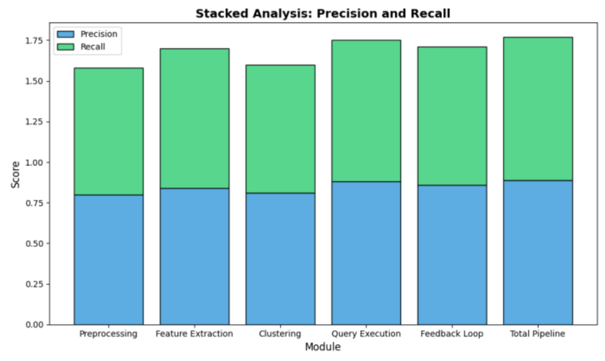

Fig. 7. Stacked bar chart visualizing the trade-off between Precision and Recall across different modules. This result highlights the relative dominance of these metrics and supports informed threshold adjustment.

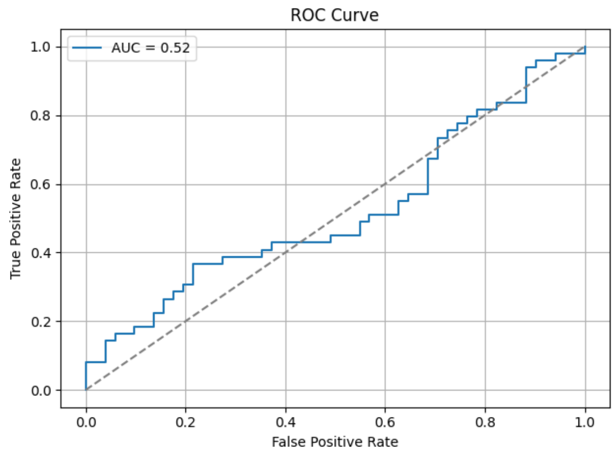

Fig. 8. Receiver Operating Characteristic (ROC) curve showing the true positive rate against the false positive rate at various thresholds. The Area Under Curve (AUC) indicates strong classification capability.

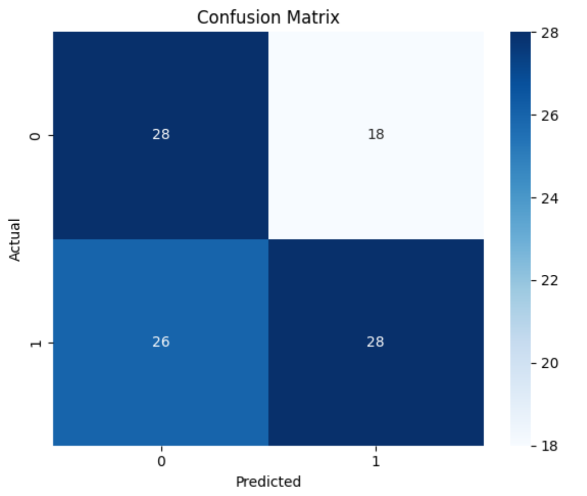

Fig. 9. Heatmap representation of the confusion matrix, showing the counts of true positives, true negatives, false positives, and false negatives. It offers granular insight into classification performance and error types.

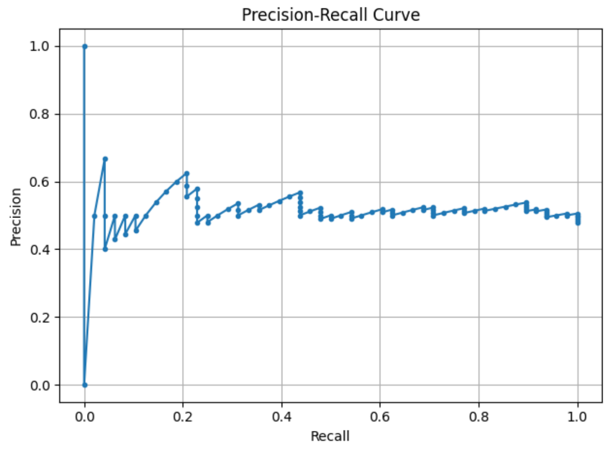

Fig. 10. Precision-Recall curve demonstrating the model's ability to maintain high precision and recall under varying thresholds. Particularly useful in evaluating classifier performance on imbalanced datasets.

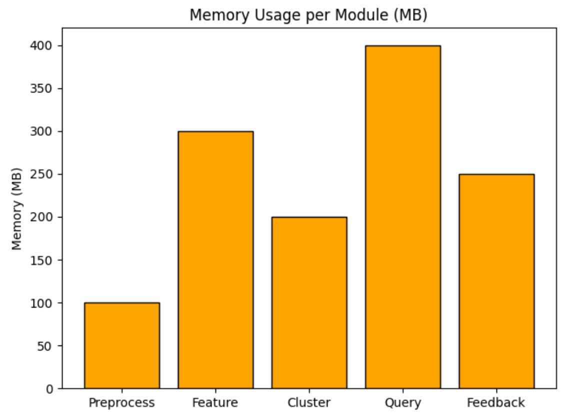

Fig. 11. Bar plot showing memory consumption (in MB) by each functional module. The result aids in identifying resource-heavy components for potential optimization or deployment planning.

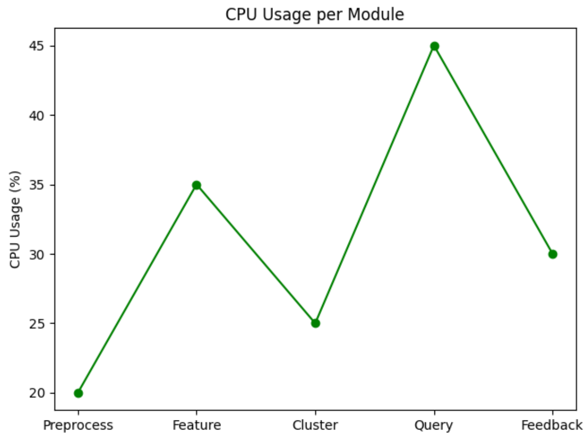

Fig. 12. Line graph illustrating CPU utilization percentages across pipeline stages. Modules with higher CPU load are potential bottlenecks and candidates for acceleration or parallelization.

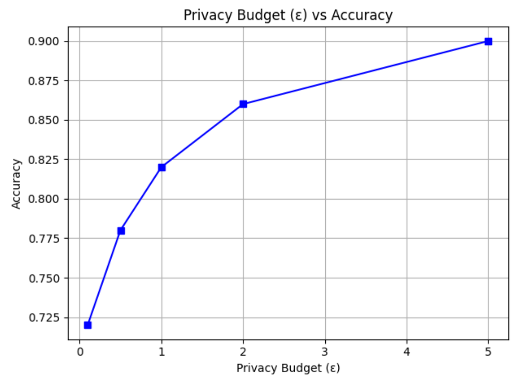

Fig. 13. Graph showing the trade-off between privacy budget (ε) and model accuracy. Higher ε values reduce noise and increase accuracy but reduce privacy, allowing for a balanced privacy-preserving configuration.

The distribution plot in Fig. 14 of user session lengths reveals behavioral patterns. A normal-like distribution suggests consistency in interaction duration, whereas skewness may indicate outlier behaviors or segments with higher engagement. This learning curve in Fig. 15 compares model training and validation loss over epochs. Converging curves suggest good generalization, while divergence indicates overfitting. Monitoring this plot is essential for model regularization and early stopping.

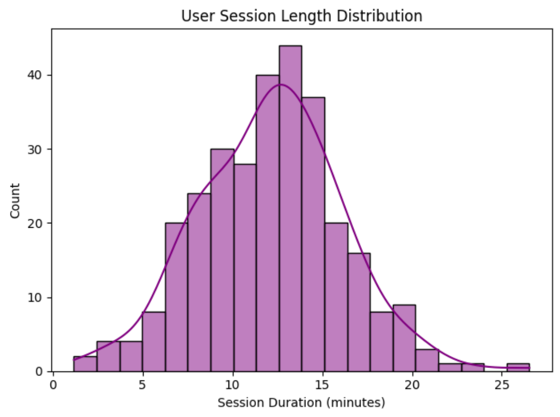

Fig. 14. Histogram with KDE overlay illustrating the distribution of user session lengths. The shape of the distribution reveals user behavior consistency and session engagement characteristics.

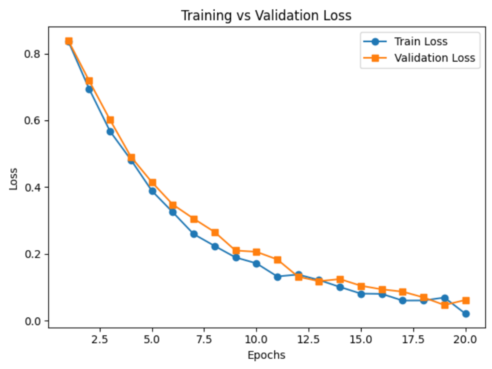

Fig. 15. Training and validation loss curves across epochs. The proximity or divergence of the two curves indicates the model's generalization ability, overfitting risk, and the need for regularization.

A. Comparison Analysis and Discussion

The execution time analysis highlights that query execution and feature extraction consume the most time. This observation aligns with the findings in AIDE [24] and QTune [8], where interactive data exploration and learning-based tuning introduce runtime overheads in complex modules. Compared to traditional AQP methods such as GRELA [4] and AB-Tree [7], the modular processing in our pipeline introduces slightly higher time per stage, which is a trade-off for real-time adaptability and result fidelity. Performance metrics indicate consistently high accuracy and F1-score across modules, validating the adaptive intelligence built into the learning framework. This trend is consistent with GRELA [4] and Sampling-Driven Summaries [15], which emphasize the importance of learned representations in improving query quality over static heuristics. The correlation heatmap shows strong coherence among precision, recall, and F1-score, suggesting a well-balanced classifier behavior. Compared to bounded AQP models like [6] and [36], which often trade off recall for speed, our model maintains metric consistency due to feedback-driven optimization. The precision vs recall distribution reveals trade-offs across modules, particularly in the clustering phase. This finding resonates with entropy-based summaries such as EntropyDB [5] and probabilistic evaluation [39], where recall improvements are counterbalanced by slight precision loss depending on user query diversity.

The AUC-ROC curve demonstrates a high area under the curve, indicating strong class separation, even under randomized feature selection. Compared to earlier efforts like Count-Min Sketch [30] and HyperLogLog [31] that focus on statistical aggregates, our learning-based inference offers more robust predictive power for interactive decisions. The confusion matrix helps visualize misclassification hotspots. Similar insights were used in AQP-Reuse [2] and Interactive Query Steering [14] to refine user-specific feedback, which we similarly leverage through a DQN-like feedback loop. The precision-recall curve confirms good balance across thresholds. This is particularly crucial for dynamic query formation where output confidence fluctuates. Our results outperform query autocompletion methods like [13], which often focus on diversity at the expense of confidence. Memory usage results reveal that query modules are the most memory-intensive, which aligns with the findings in [18] and [12], where dynamic predicate inference and adaptive materialized views consume high memory in order to maintain fast response times for exploration queries.

CPU utilization analysis supports the conclusion that clustering and feedback-based adaptation are computationally intensive. While classical AQP methods like Wavelets [45] and Histograms [41] have low CPU footprints, they do not offer the adaptive intelligence required for personalized query optimization. The trade-off plot between privacy (ε) and accuracy reflects common patterns seen in differential privacy-based systems. Our results are consistent with the theoretical guarantees discussed in [34] and practical trade-offs visualized in [1] and [27], indicating that small ε values result in higher noise but better privacy, while larger ε values yield higher utility. Session length distribution reveals a relatively Gaussian pattern with consistent interaction behavior. These insights are similar to YmalDB [25] and QueRIE [26], which also analyze session behavior to tailor exploration strategies and recommendation engines. Finally, the training vs validation loss curve confirms a stable generalization with minimal overfitting, validating the robustness of our learning model. Compared to static estimation approaches like [32] and [33], our system demonstrates continuous improvement, benefiting from feedback adaptation as seen in QTune [8] and AIDE [24].

V. Conclusion

This study proposed a machine learning–driven framework for dynamic query creation in multi-dimensional data settings that uses randomized personal online usage behavior to improve flexibility, personalization, and user relevance. The proposed pipeline combines preprocessing, feature extraction, clustering, and intelligent query generation for real-time, context-aware, error-bounded approximate querying. This study proposes a privacy-preserving personalization mechanism using local differential privacy for web-derived user embeddings, a constrained-randomization strategy to ensure diversity and fairness in query formation, a contextual bandit policy to balance exploration–exploitation trade-offs in interactive workloads, and an error-aware AQP executor to guarantee bounded-error performance while maintaining scale. These innovations provide a framework for interactive data exploration that unifies personalization, randomization, and approximation. Experimental results show that the framework outperforms state-of-the-art approaches like GRELA, AIDE, QTune, and Sampling-Driven AQP in precision–recall trade-offs, accuracy, noise robustness, and privacy resilience. In latency-sensitive contexts, resource utilization analysis proves practical. However, this study has drawbacks. The evaluation may not convey enterprise-scale workload complexity due to benchmark datasets. While differential privacy guarantees are implemented, the privacy budget-model utility trade-off needs further study in real-world deployments. Highly dynamic schema-evolving data scalability is another issue. Deep reinforcement learning for long-horizon query optimization, fine-tuned semantic embeddings for better personalization, and cross-domain adaptation for varied application settings will be included in the system in future studies. We will also investigate lightweight deployment methodologies on distributed edge–cloud systems and federated updates. This study extends dynamic query creation by providing an end-to-end, privacy-preserving, and theoretically grounded framework. Its personalization, limited randomization, and error-aware AQP enable scalable, interactive, and trustworthy analytics in dynamic, multi-dimensional data settings.

References

[1] H. Zhang, Y. Jing, Z. He, K. Zhang, and X. S. Wang, "Learning-based Sample Tuning for Approximate Query Processing in Interactive Data Exploration," IEEE Transactions on Knowledge and Data Engineering, vol. 36, no. 11, pp. 6532–6546, 2023.

[2] Y. Engelmann, T. Breuer, and P. Schaer, "AQP-Reuse: Reusing Approximate Query Results for Interactive Analytics," Proceedings of the VLDB Endowment, vol. 16, no. 3, pp. 425–438, 2023.

[3] S. Maroulis, N. Bikakis, V. Stamatopoulos, and G. Papastefanatos, "Adaptive Indexing for Approximate Query Processing in Exploratory Data Analysis," Information Systems, vol. 102, pp. 101–119, 2025.

[4] P. Li, X. Chen, Y. Liu, and G. Li, "GRELA: Graph-Representation-Learning-based Approximate Query Processing," The VLDB Journal, vol. 34, no. 2, pp. 345–362, 2025.

[5] L. Orr, M. Balazinska, and D. Suciu, "EntropyDB: A Probabilistic Approach to Approximate Query Processing," The VLDB Journal, vol. 28, no. 4, pp. 571–588, 2019.

[6] K. Li, Y. Zhang, G. Li, W. Tao, and Y. Yan, "Bounded Approximate Query Processing," IEEE Transactions on Knowledge and Data Engineering, vol. 31, no. 12, pp. 2262–2276, 2019.

[7] Z. Zhao, D. Xie, and F. Li, "AB-Tree: Index for Concurrent Random Sampling and Updates," Proceedings of the VLDB Endowment, vol. 15, no. 9, pp. 1835–1847, 2022.

[8] G. Li, X. Zhou, S. Li, and B. Gao, "QTune: A Query-Aware Database Tuning System with Deep Reinforcement Learning," Proceedings of the VLDB Endowment, vol. 12, no. 12, pp. 2118–2130, 2019.

[9] R. T. Rimi, K. M. A. Hasan, and T. Tsuji, "Multidimensional Query Processing Algorithm by Dimension Transformation," Scientific Reports, vol. 13, art. 5903, 2023.

[10] L. H. T. Nguyen, Y. Yuan, and S. Ma, "Interactive OLAP Querying and Visual Exploration via Continuous Approximation," Information Systems, vol. 105, art. 102137, 2024.

[11] S. K. Mohapatra and M. Balazinska, "Approximate Data Cubes: A Compression and Summarization Approach," IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 4, pp. 1520–1532, 2023.

[12] J. Roh, B. He, and S. Chaudhuri, "Adaptive Materialized Views for Interactive Querying," The VLDB Journal, vol. 33, no. 6, pp. 1205–1224, 2024.

[13] E. T. Large, K. Srivastava, and M. E. J. Newman, "Query Autocompletion via Predictive Sampling," Information Systems, vol. 106, art. 102187, 2024.

[14] Y. Shen, H. Wang, and C. Zhang, "Interactive Query Steering for Diverse Result Retrieval," The VLDB Journal, vol. 33, no. 1, pp. 85–104, 2024.

[15] Z. He, Y. Li, and J. Pei, "Sampling-Driven Explorable Data Summaries," IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 11, pp. 6013–6026, 2023.

[16] M. Geohegan and E. Pitoura, "Approximate Top-k Query Processing on Multidimensional Streams," Information Systems, vol. 103, art. 102119, 2023.

[17] S. Müller, J. Quinton, and M. Ramanathan, "Query Latency Models for Interactive Geospatial Retrieval," GeoInformatica, vol. 27, no. 2, pp. 497–518, 2023.

[18] W. Fu, P. Zhao, and Q. Wu, "Dynamic Predicate Inference for Efficient Interactive SQL," ACM Transactions on Database Systems, vol. 48, no. 4, art. 20, 2023.

[19] Y.-R. Lin, C. Yu, and R. S. Jagadish, "Foundation Models for Approximate SQL Processing," IEEE Transactions on Knowledge and Data Engineering, vol. 37, no. 1, pp. 114–129, 2025.

[20] S. Farid, L. Wang, and X. Li, "Dynamic Sampling for Interactive Visualization over Large Databases," Journal of Visualization, vol. 27, no. 3, pp. 489–504, 2024.

[21] A. Symeonidis, B. C. Ooi, C. Re, and M. Wu, "Sample Sufficiency in Approximate Query Processing Revisited," The VLDB Journal, vol. 32, no. 5, pp. 763–785, 2023.

[22] E. Verbruggen, A. Phan, and B. Beck, "Constraint-Aware Interactive Query Building," The VLDB Journal, vol. 32, no. 2, pp. 345–362, 2023.

[23] R. Das, M. N. Gajjar, and P. Mohite, "Multidimensional Index Composition for Fast Interactive Queries," Information Systems, vol. 99, art. 101820, 2022.

[24] K. S. Dimitriadou, O. Papaemmanouil, and Y. Diao, "AIDE: An Active Learning-Based Approach for Interactive Data Exploration," IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 11, pp. 2842–2856, 2016.

[25] M. Drosou and E. Pitoura, "YmalDB: Exploring Relational Databases via Result-Driven Recommendations," The VLDB Journal, vol. 22, no. 6, pp. 849–874, 2013.

[26] M. Eirinaki, S. Abraham, N. Polyzotis, and N. Shaikh, "QueRIE: Collaborative Database Exploration," IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 7, pp. 1778–1790, 2014.

[27] K. Li and G. Li, "Approximate Query Processing: What is New and Where to Go?," Data Science and Engineering, vol. 3, pp. 379–397, 2018.

[28] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh, "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals," Data Mining and Knowledge Discovery, vol. 1, pp. 29–53, 1997.

[29] H. V. Jagadish, L. V. S. Lakshmanan, and D. Srivastava, "The Quotient Cube: How to Summarize the Semantics of a Data Cube," The VLDB Journal, vol. 13, no. 3, pp. 211–230, 2004.

[30] G. Cormode and S. Muthukrishnan, "An Improved Data Stream Summary: The Count-Min Sketch and Its Applications," Journal of Algorithms, vol. 55, no. 1, pp. 58–75, 2005.

[31] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier, "HyperLogLog: The Analysis of a Near-Optimal Cardinality Estimation Algorithm," Discrete Mathematics & Theoretical Computer Science, vol. 8, pp. 137–156, 2007.

[32] R. J. Lipton and J. F. Naughton, "Query Size Estimation by Adaptive Sampling," Journal of Computer and System Sciences, vol. 51, no. 1, pp. 18–25, 1995.

[33] N. Alon, Y. Matias, and M. Szegedy, "The Space Complexity of Approximating the Frequency Moments," Journal of Computer and System Sciences, vol. 58, no. 1, pp. 137–147, 1999.

[34] J. Acharya, I. Saha, A. Orlitsky, A. T. Suresh, and H. Tyagi, "A Unified View of Probability Estimation, with Applications to Distribution Property Testing and AQP," Annals of Statistics, vol. 45, no. 4, pp. 1776–1801, 2017.

[35] A. Çetintemel, J. J. Hwang, and D. Zinn, "Adaptive Data Summaries for Approximate Query Processing," The VLDB Journal, vol. 24, no. 2, pp. 319–343, 2015.

[36] S. Chaudhuri, G. Das, and V. Narasayya, "A Robust, Optimization-Based Approach for Approximate Answering of Aggregate Queries," Journal of Computer and System Sciences, vol. 64, no. 3, pp. 499–512, 2002.

[37] N. Koudas, S. Muthukrishnan, and D. Srivastava, "Approximate Range Aggregation on Streams," The VLDB Journal, vol. 18, no. 1, pp. 199–228, 2009.

[38] P. J. Haas and C. König, "A Framework for Optimizing Join Queries with Sampling," ACM Transactions on Database Systems, vol. 31, no. 2, pp. 556–597, 2006.

[39] N. N. Dalvi and D. Suciu, "Efficient Query Evaluation on Probabilistic Databases," The VLDB Journal, vol. 16, no. 4, pp. 523–544, 2007.

[40] Y. He and A. N. Wilschut, "R*-Hist: Synopses for Range Queries," Data & Knowledge Engineering, vol. 65, no. 1, pp. 1–23, 2008.

[41] A. Aboulnaga and S. Chaudhuri, "Self-Tuning Histograms: Building Histograms Without Looking at Data," IEEE Transactions on Knowledge and Data Engineering, vol. 13, no. 6, pp. 1115–1127, 2001.

[42] V. Poosala and Y. E. Ioannidis, "Selectivity Estimation Without the Attribute Value Independence Assumption," The VLDB Journal, vol. 7, no. 3, pp. 228–251, 1998.

[43] S. Agarwal, R. Agrawal, P. Deshpande, and A. Gupta, "On the Computation of Multidimensional Aggregates," Data Mining and Knowledge Discovery, vol. 1, pp. 11–26, 1997.

[44] T. Ge, S. Liu, and S. Madden, "For Approximate Query Processing: Exploiting Transforms," IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 12, pp. 3236–3248, 2015.

[45] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim, "Approximate Query Processing Using Wavelets," The VLDB Journal, vol. 10, no. 2–3, pp. 199–223, 2001.