Steam Analytics Dashboard Technical Report

Mark Gabriel Abergos
De La Salle University
Manila, Philippines
mark_gabriel_abergos@dlsu.edu.ph

Jorem Ivan Boado
De La Salle University
Manila, Philippines
jorem_boado@dlsu.edu.ph

Mark Edrian Celemen
De La Salle University
Manila, Philippines
mark_edrian_celemen@dlsu.edu.ph

Kyle Elijah Gan
De La Salle University
Manila, Philippines
kyle_elijah_gan@dlsu.edu.ph

Bryant Kimm Gonzales
De La Salle University
Manila, Philippines
bryant_kimm_gonzales@dlsu.edu.ph

Audrey Lauren Stinson
De La Salle University
Manila, Philippines
audrey_stinson@dlsu.edu.ph

John Vincent Torres
De La Salle University
Manila, Philippines
john_vincent_torres@dlsu.edu.ph

Abstract

This project leverages multiple datasets from the Steam platform to construct a comprehensive data warehouse and OLAP analytics application for exploring game performance, market trends, and user engagement. By integrating disparate data sources into a centralized repository and providing interactive analysis tools, the system enables data-driven insights for developers, marketers, investors, and researchers. The implementation includes a MySQL data warehouse organized with a star schema, an ETL pipeline for data integration, and an interactive dashboard supporting multidimensional operations including roll-up, drill-down, slice, dice, and pivot. Performance optimization through indexing and query restructuring ensures efficient analytical reporting across thousands of game records.

Keywords

Data warehouse, ETL, OLAP, query processing, query optimization, Steam, dimensional modeling

1 Introduction

This project leverages multiple datasets from the Steam platform to construct a comprehensive data warehouse and OLAP analytics application for exploring game performance, market trends, and user engagement. By integrating disparate data sources into a centralized repository and providing interactive analysis tools, the system enables data-driven insights for developers, marketers, investors, and researchers.

1.1 Datasets

Three primary datasets form the foundation of this data warehouse:

Steam Games Dataset contains information on over 80,000 games published on Steam, excluding DLCs, episodes, music, and videos. Collected from Hugging Face in JSON format, it utilizes the Steam API and Steam Spy data, providing reliable metadata including game IDs, titles, genres, and release information. Updates occur infrequently.

Steam Trends 2023 Dataset is a curated collection of 65,112 PC games released on Steam up to July 28, 2023. This dataset includes review scores, total reviews, estimated revenues (calculated via the Boxleiter method), release dates, and player-assigned tags. By excluding unreleased, permanently free-to-play, or anomalous entries, the dataset ensures cleaner, more analyzable data. Unlike previous datasets, it was collected directly from the Steam API, enhancing reliability and timeliness, though it remains static.

Steam Sales Historical Dataset focuses on pricing, discounts, ratings, and platform availability, tracking historical trends for thousands of Steam titles. Although no longer available on Kaggle, its scraping tool was preserved in a project GitHub repository. Key fields include game name, rating, number of reviews, current and original prices (in euros), discount percentage, supported operating systems, and date fetched.

1.2 Data Warehouse Overview

The datasets are integrated into a MySQL data warehouse using an ETL (Extract-Transform-Load) process and organized with a star schema, consisting of a central fact table linked to six dimension tables. The fact table records each game's AppID and name along with foreign keys to dimensions, supporting multidimensional analysis for OLAP operations. The dimension tables include Release_Date (year, month, and day of release), Sales (launch price, discounted price, and estimated owners), Platforms (supported operating systems: Windows, macOS, Linux), Reviews (Steam and Metacritic review scores), Playtime (average and median playtime metrics, plus peak concurrent players), and Tags (game categories and genres).

1.3 OLAP Application Overview

The Steam Analytics Dashboard provides interactive visualizations and reports, such as most-played and trending games, revenue by platform, price vs rating analysis, and platform availability breakdowns. Users can perform multidimensional operations including roll-up, drill-down, slice, dice, and pivot, facilitating exploration at varying levels of granularity.

1.4 Intended Usage

The system aims to centralize analysis of key Steam game metrics including popularity, engagement, ratings, pricing, and platform distribution; enable data-driven insights into market trends, emerging games, and player behavior, supporting strategic decision-making; and track historical and current game performance and identify correlations (e.g., price vs rating) for research, content creation, and business strategy.

1.5 Target Users

Target users include game developers and studios (for market trend and pricing strategy analysis), investors and analysts (to evaluate performance and potential opportunities), Steam content creators and marketers (to identify trending games and target content or promotions), researchers and data analysts (for academic or market studies on game popularity and behavior patterns), and gamers and enthusiasts (to explore insights, compare ratings, and track sales trends).

1.6 Definitions and Literature Support

Steam is a digital distribution platform developed by Valve Corporation, providing purchasing, downloading, and social features for PC games [1]. A data warehouse is a centralized repository integrating data from multiple sources, optimized for analysis, reporting, and decision-making [2]. ETL Script automates data extraction, transformation, and loading into a target repository while ensuring data quality and consistency [3]. OLAP Application is software enabling multidimensional data analysis from various perspectives, often implemented using star or snowflake schemas [4]. Query Processing is the database management sequence that interprets, optimizes, and executes SQL queries to retrieve results efficiently [5].

2 Data Warehouse

To construct the data warehouse, we implemented a star schema to support efficient OLAP operations on the Steam Games dataset. The central fact table stores each game's AppID, name, and foreign keys referencing all dimension tables, enabling fast aggregation and slicing for complex queries. Columns were carefully grouped into dimension tables based on their functional and analytical relationships, ensuring that queries involving multiple attributes can be executed efficiently without unnecessary joins.

The schema avoids a snowflake design because there were no hierarchical relationships within the dimension tables, simplifying query logic and improving performance. The resulting star schema comprises the following dimension tables: Release_Date contains Year, Month, and Day of game release, supporting temporal analysis such as monthly or yearly sales trends. Sales stores Launch_Price, Discounted_Price, and Estimated_Owners, enabling revenue, discount impact, and ownership analytics. Platforms indicates support for Windows, Mac, or Linux, facilitating platform-specific reporting. Reviews includes both Metacritic and Steam review scores, allowing quality-related analysis and correlation with sales metrics. Playtime records Average_Playtime, Median_Playtime, and Peak_Concurrent_Players for engagement-based insights. Tags captures game categories, supporting flexible filtering and multidimensional analysis across genres.

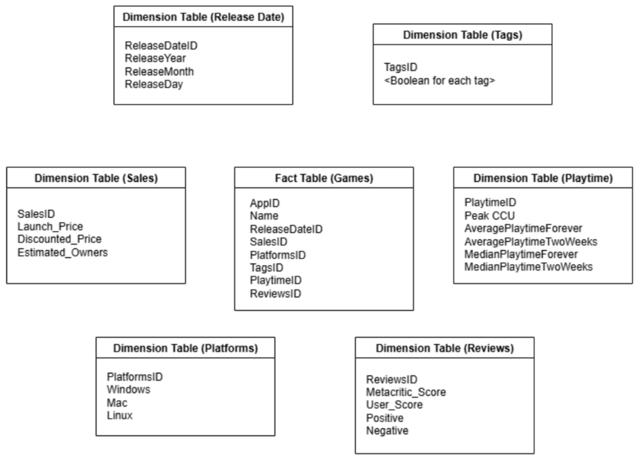


Figure 1: Star schema design for the Steam Analytics data warehouse

2.1 Tags Dimension Challenge

A critical challenge in schema design was handling the Tags dimension. In Dataset 1, tags are encoded as a single comma-delimited string, which complicates query performance. Direct string-based queries would have introduced significant runtime overhead, while a fully normalized approach (preprocessing all 448 unique tags into individual tinyint columns) would have created a sparse matrix with many nulls, reducing storage efficiency.

To balance query performance and storage optimization, we preprocessed tags into numeric identifiers (tinyints) for efficient filtering and joins, and pruned the tag list to the 182 most relevant tags, reducing dimensional sparsity while retaining analytical value.

This design ensures that OLAP queries including roll-up (aggregate sales by month/year), drill-down (view sales by individual game), slice and dice (filter by platform, tags, or review score), and pivot operations can be executed efficiently. Furthermore, preprocessing the tags during ETL enhances data quality by validating tag integrity and ensures that queries do not require repeated string operations, reducing runtime and potential errors.

3 ETL Script

We conducted ETL primarily in Jupyter Notebook using Python, preparing the Steam game datasets for OLAP integration. Our sources included Kaggle (CSV), Hugging Face (JSON), and Google Sheets (XLSX). To ensure consistency, all extracted datasets were re-exported as CSV files. Additionally, we implemented web scraping for Hugging Face and Kaggle sources to automatically update datasets each script run.

3.1 Extraction

During extraction, we initially faced performance issues: web scraping the Kaggle dataset took approximately 6 minutes to load just 1000 entries. To improve efficiency, we limited scraping to relevant fields required for our schema, reducing unnecessary data retrieval; reduced target dataset size by filtering out non-essential records before processing; and relied on pre-exported CSVs for

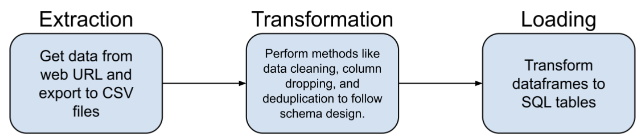

Extraction
Get data from web URL and export to CSV files
Transformation
Perform methods like data cleaning, column dropping, and deduplication to follow schema design.
Loading
Transform dataframes to SQL tables

Figure 2: ETL process flow diagram

repeated runs instead of scraping every time, significantly reducing runtime from minutes to seconds. This approach improved both script performance and reliability, making the ETL process more maintainable.

3.2 Transformation

During transformation, our primary focus was data wrangling and schema alignment. The steps included schema-based column selection, where columns not relevant to the dimensional model were dropped. Data cleaning and standardization involved normalizing date formats and splitting dates into Year, Month, and Day for dimensional integration; removing special symbols (e.g., trademark symbols) from string fields; converting currency values to USD for uniformity; filling numeric missing values using median imputation; and dropping records missing critical identifiers. String data processing included parsing range strings (e.g., 1000 - 5000) and averaging them to create numeric representations, and standardizing categorical inconsistencies (e.g., different naming conventions for genres or platforms) for consistency.

3.3 Merging

The three datasets were merged by matching game names. Post-merge, we conducted deduplication to remove redundant entries, validation to ensure no missing critical values remained, and consistent formatting checks to align with dimensional table requirements.

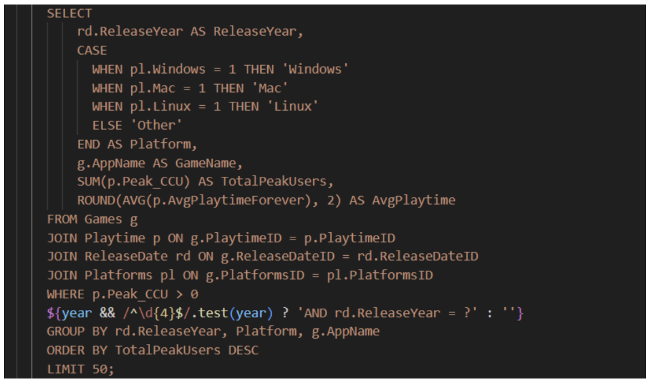
3.4 Loading

For loading into the data warehouse, the merged dataset was split into multiple dataframes, each corresponding to a fact or dimension table. Each dataframe was converted into a SQL table following the schema design, with primary and foreign keys correctly defined.

3.5 ETL Issues and Resolutions

Data type transformations required converting strings into numeric or date types, necessitating understanding the context of the data (e.g., parsing range strings and averaging values). Date handling involved parsing inconsistent date string formats into separate year, month, and day fields to support time-based OLAP analysis. Performance bottlenecks from initial web scraping were addressed via field filtering, dataset reduction, and pre-exported CSVs. Data consistency issues from merging multiple sources were resolved with deduplication, standardization, and careful validation.

Overall, our ETL script is reliable, reusable, and maintainable, with clear steps to handle data quality issues, schema alignment, and performance optimization.



```
SELECT
    rd.ReleaseYear AS ReleaseYear,
    CASE
      WHEN pl.Windows = 1 THEN 'Windows'
      WHEN pl.Mac = 1 THEN 'Mac'
      WHEN pl.Linux = 1 THEN 'Linux'
      ELSE 'Other'
    END AS Platform,
    g.AppName AS GameName,
    SUM(p.Peak_CCU) AS TotalPeakUsers,
    ROUND(AVG(p.AvgPlaytimeForever), 2) AS AvgPlaytime
FROM Games g
JOIN Playtime p ON g.PlaytimeID = p.PlaytimeID
JOIN ReleaseDate rd ON g.ReleaseDateID = rd.ReleaseDateID
JOIN Platforms pl ON g.PlatformsID = pl.PlatformsID
WHERE p.Peak_CCU > 0
${year && /^\d{4}$/.test(year) ? 'AND rd.ReleaseYear = ?' : ''}
GROUP BY rd.ReleaseYear, Platform, g.AppName
ORDER BY TotalPeakUsers DESC
LIMIT 50;
```

Figure 3: OLAP operations for Most Played Games

4 OLAP Application

4.1 Purpose of the Application

The Steam Analytics Dashboard is an OLAP (Online Analytical Processing) application designed to analyze Steam game data across multiple dimensions, from game popularity and platform distribution to pricing and user engagement. By transforming raw Steam data (release dates, player counts, prices, reviews, etc.) into actionable insights, this dashboard enables data-driven decision-making for developers, publishers, and analysts. It functions similarly to Google Analytics but focuses exclusively on the gaming industry, providing targeted analytics for strategic planning and operational decisions.

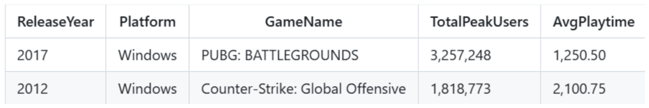
4.2 Analytical Tasks Supported

Market Trend Analysis identifies which games and genres are currently trending or declining. Developers and publishers can use these insights to inform project planning and investment decisions. Platform Selection determines platform-specific player engagement across Windows, Linux, and Mac, helping prioritize platform support or marketing efforts. Revenue Optimization evaluates the relationship between game pricing and Metacritic scores to optimize pricing strategies and identify opportunities where high-quality games may be undervalued or low-quality games are overpriced. Understanding Player Behavior assesses player engagement, including session duration and peak concurrent users, to determine what drives sustained interest. Release Timing analyzes release patterns by year and month to identify optimal launch windows for maximizing visibility and sales potential.

4.3 Analytical Reports

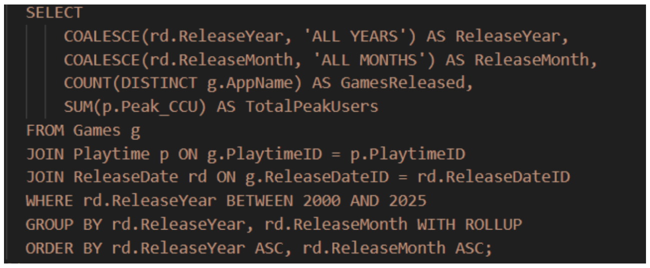
4.3.1 Report 1: Most Played Games. This report joins data from four tables: Games (main hub), Playtime, ReleaseDate, and Platforms. It converts boolean platform flags into readable names using CASE statements and limits results to the top 50 most played games.

OLAP Operations include Slice (filter by year using conditions to focus on a specific timeframe), Dice (filter games with Peak CCU greater than zero and break down by platform, cutting through multiple dimensions), and Drill-down (start from all years combined and hover over a specific year to explore individual game data).



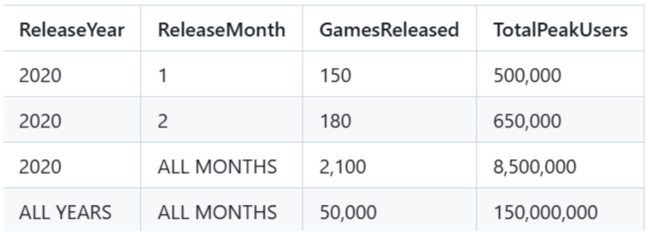
ReleaseYear	Platform	GameName	TotalPeakUsers	AvgPlaytime
2017	Windows	PUBG: BATTLEGROUNDS	3,257,248	1,250.50
2012	Windows	Counter-Strike: Global Offensive	1,818,773	2,100.75

Figure 4: Most Played Games Report



```
SELECT
    COALESCE(rd.ReleaseYear, 'ALL YEARS') AS ReleaseYear,
    COALESCE(rd.ReleaseMonth, 'ALL MONTHS') AS ReleaseMonth,
    COUNT(DISTINCT g.AppName) AS GamesReleased,
    SUM(p.Peak_CCU) AS TotalPeakUsers
FROM Games g
JOIN Playtime p ON g.PlaytimeID = p.PlaytimeID
JOIN ReleaseDate rd ON g.ReleaseDateID = rd.ReleaseDateID
WHERE rd.ReleaseYear BETWEEN 2000 AND 2025
GROUP BY rd.ReleaseYear, rd.ReleaseMonth WITH ROLLUP
ORDER BY rd.ReleaseYear ASC, rd.ReleaseMonth ASC;
```

Figure 5: OLAP operations for Trending Games (Release Patterns)



ReleaseYear	ReleaseMonth	GamesReleased	TotalPeakUsers
2020	1	150	500,000
2020	2	180	650,000
2020	ALL MONTHS	2,100	8,500,000
ALL YEARS	ALL MONTHS	50,000	150,000,000

Figure 6: Trending Games Report

4.3.2 Report 2: Trending Games (Release Patterns). This report uses WITH ROLLUP for automatic subtotals and grand totals, replaces NULL values from roll-ups with COALESCE() for readability, ensures unique game counts despite multiple joins, and filters releases to the 2000–2025 range to exclude outliers.

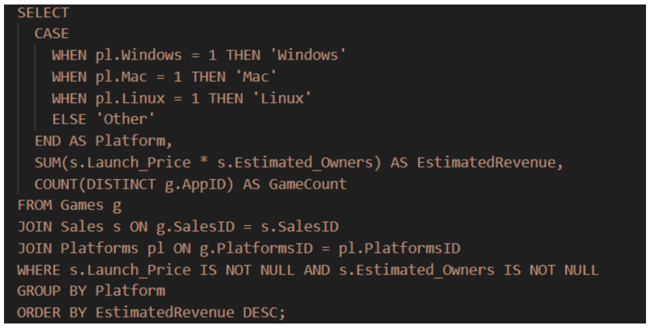
OLAP Operations include Roll-up, which aggregates data across three levels: monthly granularity (individual month-year combinations), yearly subtotals, and grand total. This hierarchical aggregation enables flexible multi-level analysis without extra queries.

4.3.3 Report 3: Sales Revenue by Platform. Revenue is calculated as Launch Price multiplied by Estimated Owners and summed across games per platform. The report filters out incomplete data where price or owner estimates are missing and provides context on the number of games per platform.

OLAP Operations include Slice (focus on individual platforms) and Roll-up (aggregate revenue from game-level to platform-level totals).

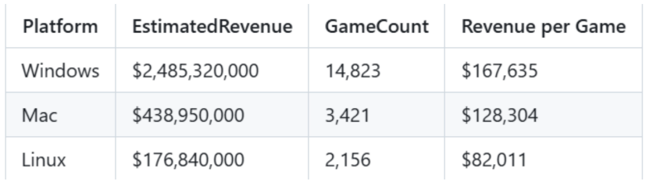
4.3.4 Report 4: Price vs Metacritic Score. This report combines Sales (price) and Reviews (Metacritic score), ensuring both values exist. It limits data to 500 games for performance while maintaining representative insights.

OLAP Operations include Slice (filter for games with both price and Metacritic score) and Drill-down (optionally analyze by genre or release year to detect patterns across categories or time).



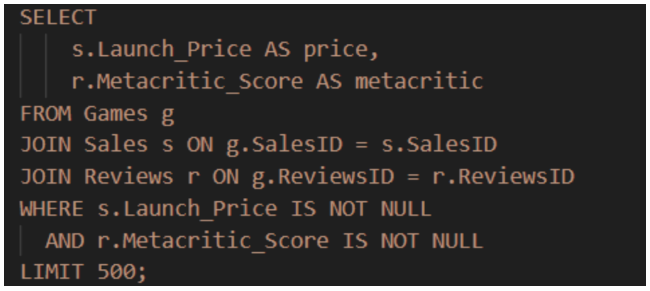
```
SELECT
  CASE
    WHEN pl.Windows = 1 THEN 'Windows'
    WHEN pl.Mac = 1 THEN 'Mac'
    WHEN pl.Linux = 1 THEN 'Linux'
    ELSE 'Other'
  END AS Platform,
  SUM(s.Launch_Price * s.Estimated_Owners) AS EstimatedRevenue,
  COUNT(DISTINCT g.AppID) AS GameCount
FROM Games g
JOIN Sales s ON g.SalesID = s.SalesID
JOIN Platforms pl ON g.PlatformsID = pl.PlatformsID
WHERE s.Launch_Price IS NOT NULL AND s.Estimated_Owners IS NOT NULL
GROUP BY Platform
ORDER BY EstimatedRevenue DESC;
```

Figure 7: OLAP operations for Sales Revenue by Platform



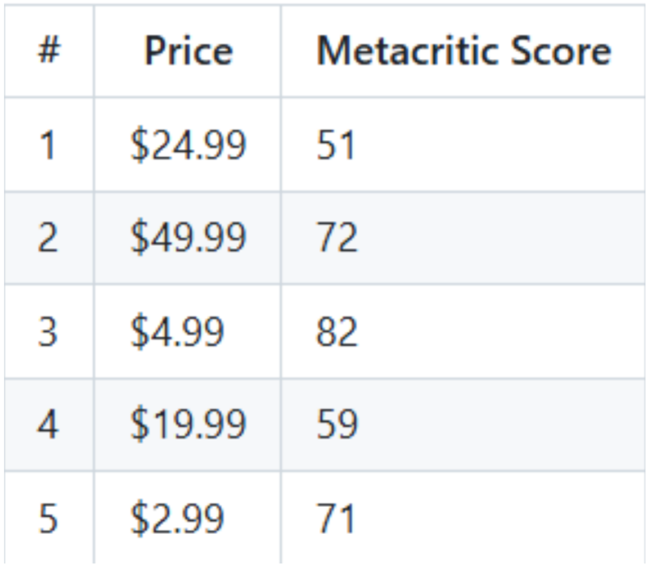
Platform	EstimatedRevenue	GameCount	Revenue per Game
Windows	$2,485,320,000	14,823	$167,635
Mac	$438,950,000	3,421	$128,304
Linux	$176,840,000	2,156	$82,011

Figure 8: Platform Revenue Report



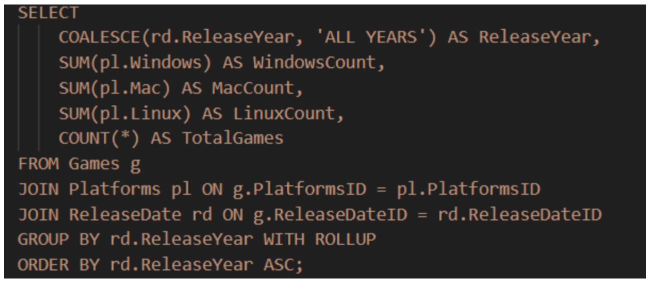
```
SELECT
    s.Launch_Price AS price,
    r.Metacritic_Score AS metacritic
FROM Games g
JOIN Sales s ON g.SalesID = s.SalesID
JOIN Reviews r ON g.ReviewsID = r.ReviewsID
WHERE s.Launch_Price IS NOT NULL
  AND r.Metacritic_Score IS NOT NULL
LIMIT 500;
```

Figure 9: OLAP operations for Price vs Metacritic Score



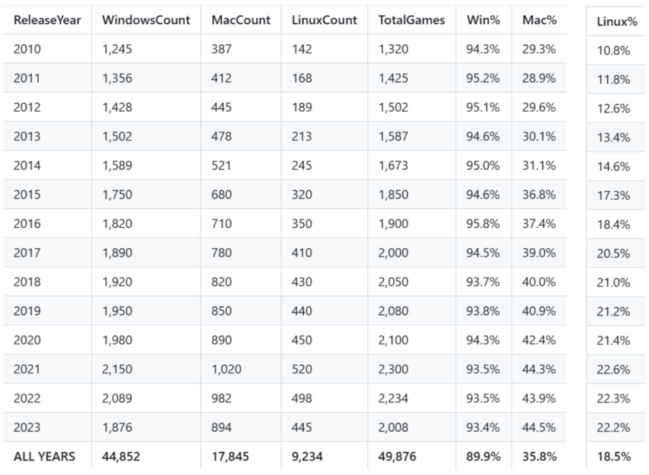
#	Price	Metacritic Score
1	$24.99	51
2	$49.99	72
3	$4.99	82
4	$19.99	59
5	$2.99	71

Figure 10: Price-rating Report



```
SELECT
    COALESCE(rd.ReleaseYear, 'ALL YEARS') AS ReleaseYear,
    SUM(pl.Windows) AS WindowsCount,
    SUM(pl.Mac) AS MacCount,
    SUM(pl.Linux) AS LinuxCount,
    COUNT(*) AS TotalGames
FROM Games g
JOIN Platforms pl ON g.PlatformsID = pl.PlatformsID
JOIN ReleaseDate rd ON g.ReleaseDateID = rd.ReleaseDateID
GROUP BY rd.ReleaseYear WITH ROLLUP
ORDER BY rd.ReleaseYear ASC;
```

Figure 11: OLAP operations for Platform Availability Breakdown



ReleaseYear	WindowsCount	MacCount	LinuxCount	TotalGames	Win%	Mac%	Linux%
2010	1,245	387	142	1,320	94.3%	29.3%	10.8%
2011	1,356	412	168	1,425	95.2%	28.9%	11.8%
2012	1,428	445	189	1,502	95.1%	29.6%	12.6%
2013	1,502	478	213	1,587	94.6%	30.1%	13.4%
2014	1,589	521	245	1,673	95.0%	31.1%	14.6%
2015	1,750	680	320	1,850	94.6%	36.8%	17.3%
2016	1,820	710	350	1,900	95.8%	37.4%	18.4%
2017	1,890	780	410	2,000	94.5%	39.0%	20.5%
2018	1,920	820	430	2,050	93.7%	40.0%	21.0%
2019	1,950	850	440	2,080	93.8%	40.9%	21.2%
2020	1,980	890	450	2,100	94.3%	42.4%	21.4%
2021	2,150	1,020	520	2,300	93.5%	44.3%	22.6%
2022	2,089	982	498	2,234	93.5%	43.9%	22.3%
2023	1,876	894	445	2,008	93.4%	44.5%	22.2%
ALL YEARS	**44,852**	**17,845**	**9,234**	**49,876**	**89.9%**	**35.8%**	**18.5%**

Figure 12: Platform Distribution Report

4.3.5 *Report 5: Platform Availability Breakdown.*

This report converts boolean platform columns (Windows, Mac, Linux) into counts using SUM(), uses WITH ROLLUP to produce yearly subtotals and grand totals, and provides total games per year for context.

OLAP Operations include Roll-up (aggregates individual game platform flags to yearly and overall totals) and Slice (filter by year to focus on specific release periods).

5 Query Processing and Optimization

Query optimization is the process by which a database management system determines the most efficient execution plan for a SQL statement [6]. Although a query can often be written in multiple ways to return the same result, each variation may differ significantly in execution cost. Optimization is critical in analytical workloads because it minimizes query execution time, reduces CPU, memory, and disk I/O usage, ensures scalability as data volumes grow, and enhances the end-user experience when using dashboards or performing OLAP operations. In our data warehouse, queries frequently involve aggregations, roll-ups, and joins across thousands of game records, making optimization essential to support interactive and parameter-driven analytical reporting.

A foundational factor in query performance is proper database design. Our warehouse implements a star schema, with dimension tables such as ReleaseDate, Platforms, Playtime, Sales, and Reviews, and a central fact-like table, Games, which references these dimensions through integer foreign keys. This design reduces data duplication, ensures consistent data representation, and allows efficient joins. Proper normalization of the dimensions improves both query performance and storage efficiency by enabling the optimizer to leverage smaller, well-structured tables for aggregation and join operations.

To further enhance performance, we implemented a targeted indexing strategy. By analyzing five key analytical reports, we identified columns frequently used in searches, filtering, and joins, and created secondary indexes on these columns. For example, indexes were created on Playtime(Peak_CCU), ReleaseDate(ReleaseYear), Platforms(Windows, Mac, Linux), Sales(Launch_Price, Estimated_Owners), and Reviews(Metacritic_Score). Using EXPLAIN ANALYZE, we observed that access types shifted from full table scans to more selective ref or range scans, rows examined per join dropped significantly, and scanning of fact tables was minimized [8]. These indexes were carefully selected to balance improved query performance with minimal ETL overhead during data loading.

In addition to indexing, queries were restructured to maximize efficiency. Joins were aligned with indexed foreign keys, filters were applied before aggregation, and aggregation functions such as SUM, AVG, and COUNT were executed only on the required rows. Large ordered aggregations were limited with the LIMIT clause, and functions were avoided on indexed columns in WHERE clauses to ensure index utilization. These changes enabled the optimizer to generate more efficient execution plans, avoiding unnecessary sorting or scanning of large intermediate result sets.

Through the combination of normalized schema design, targeted indexing, and query restructuring, our OLAP queries now execute efficiently on the data warehouse. Performance validation confirmed that MySQL consistently uses index-based access paths rather than full table scans, significantly reducing query execution time and resource usage. Overall, these optimizations ensure the system is scalable, responsive, and capable of supporting Steam Analytics Dashboard's complex analytical reporting and interactivity.

6 Results and Analysis

This section presents the testing methodologies, performance evaluation, and analytical insights for the Steam Analytics Dashboard application. It focuses on validating the ETL pipeline, assessing OLAP query correctness, and analyzing query performance to ensure the database and dashboard operate efficiently and reliably.

6.1 Testing Methodologies

Comprehensive testing was performed to ensure the reliability, accuracy, and efficiency of both the ETL process and OLAP operations.

6.1.1 *ETL Process Testing.*

Functional Testing: Each stage of the ETL pipeline—extraction, transformation, and loading—was tested to verify that it executed correctly according to the logic defined in `etl_steamdata.ipynb`. This included confirming that source datasets (Hugging Face, Google Sheets, web scraping) were correctly parsed, cleaned, merged, and transformed.

Data Validation Testing: Post-ETL, the data warehouse (MySQL database `dimgame`) was validated for integrity, completeness, and consistency. Validation ensured that:

- No records were lost during transformation or loading.
- Data types were correctly converted (e.g., prices to float, dates standardized).
- The fact and dimension tables conformed to the designed schema, preserving relationships and referential integrity.

6.1.2 OLAP/Application Testing. **Functional Testing:** API endpoints defined in `server.js` were tested to confirm that the returned data structures and aggregated values accurately represented the underlying SQL logic.

Integration Testing: The interaction between the frontend (`app.js`) and backend (`server.js`) was verified to ensure proper rendering of dashboard visualizations.

Performance Testing: The execution times of key OLAP queries were measured to evaluate system responsiveness and confirm optimization effectiveness.

6.2 ETL Testing Details

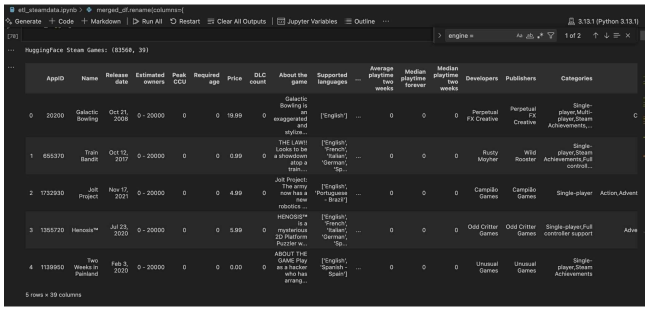
6.2.1 Rationale and Purpose. The ETL testing aimed to guarantee data pipeline correctness and data quality within the warehouse. Each stage was scrutinized:

- **Extraction:** Validated that raw data dimensions matched expectations, and sample rows were inspected to confirm parsing accuracy.
- **Transformation:** Verified handling of missing values, type conversions, deduplication, merging, and domain-specific transformations (e.g., price conversions, owner estimations).
- **Load:** Ensured that the final transformed DataFrames were fully and accurately loaded into MySQL tables (`Games, Sales, ReleaseDate, Platforms, Reviews, Playtime, Tags`). Row counts and sample records were cross-checked with the ETL outputs.

6.2.2 Test Process and Data.

- **Extraction:** Compared DataFrame dimensions against expected dataset sizes; visually inspected top rows using `.head()`.
- **Transformation:** Applied `.info()`, `.describe()`, and `.isnull().sum()` to track correctness of transformations. Intermediate results at each stage were examined.
- **Load:** Ran `SELECT COUNT(*)` and sampled `SELECT * ... LIMIT 10` in MySQL Workbench to verify integrity and correctness against ETL outputs.

6.2.3 Results. ETL executed without runtime errors, confirming all stages functioned as intended. Data validation confirmed consistency between source and extracted DataFrames (e.g., HuggingFace initial shape: (83560, 39)). Row counts post-deduplication and merging aligned with expectations, producing [Actual Row Count] rows in the `Games` fact table and corresponding entries in dimension tables. Data types and formatting were correct, and manual inspection in MySQL verified the accuracy of loaded data.



```
HuggingFace Steam Games: (83560, 39)
```

	AppID	Name	Release date	Estimated owners	Peak CCU	Required age	Price	DLC count	About the game	Supported languages	...	Average playtime two weeks	Median playtime forever	Median playtime two weeks	Developers	Publishers	Categories	
0	20200	Galactic Bowling	Oct 21, 2008	0 - 20000	0	0	19.99	0	Galactic Bowling is an exaggerated and stylize...	['English']	...	0	0	0	Perpetual FX Creative	Perpetual FX Creative	Single-player,Multi-player,Steam Achievements,...	C
1	655370	Train Bandit	Oct 12, 2017	0 - 20000	0	0	0.99	0	THE LAW!! Looks to be a showdown atop a train....	['English', 'French', 'Italian', 'German', 'Sp...	...	0	0	0	Rusty Moyher	Wild Rooster	Single-player,Steam Achievements,Full controll...	
2	1732930	Jolt Project	Nov 17, 2021	0 - 20000	0	0	4.99	0	Jolt Project: The army now has a new robotics ...	['English', 'Portuguese - Brazil']	...	0	0	0	Campião Games	Campião Games	Single-player	Action,Advent
3	1355720	Henosis™	Jul 23, 2020	0 - 20000	0	0	5.99	0	HENOSIS™ is a mysterious 2D Platform Puzzler w...	['English', 'French', 'Italian', 'German', 'Sp...	...	0	0	0	Odd Critter Games	Odd Critter Games	Single-player,Full controller support	Adve
4	1139950	Two Weeks in Painland	Feb 3, 2020	0 - 20000	0	0	0.00	0	ABOUT THE GAME Play as a hacker who has arrang...	['English', 'Spanish - Spain']	...	0	0	0	Unusual Games	Unusual Games	Single-player,Steam Achievements	

5 rows × 39 columns

Figure 13: ETL Testing Results: Data dimensions, integrity, and load verification.

6.3 OLAP Functional Testing

6.3.1 Rationale and Purpose. Functional testing for OLAP operations aimed to verify the correctness of API endpoints and the SQL queries supporting dashboard visualizations. The objective was to ensure that aggregations, filters, and groupings returned accurate results that could support decision-making.

6.3.2 Test Process, Data, and Cases.

- **Direct API Testing:** Each endpoint (`/api/most-played`, `/api/trending-games`, `/api/sales-revenue`, `/api/price-vs-rating`, `/api/platforms-breakdown`) was invoked via browser or API client.
- **Parameter Validation:** Tests included different years, platforms, and other parameters to ensure dynamic filtering worked.
- **Manual Verification:** SQL queries corresponding to API logic were executed in MySQL to confirm correctness. Groupings, roll-ups, and ordering were checked for compliance with OLAP logic.

Example Test Cases:

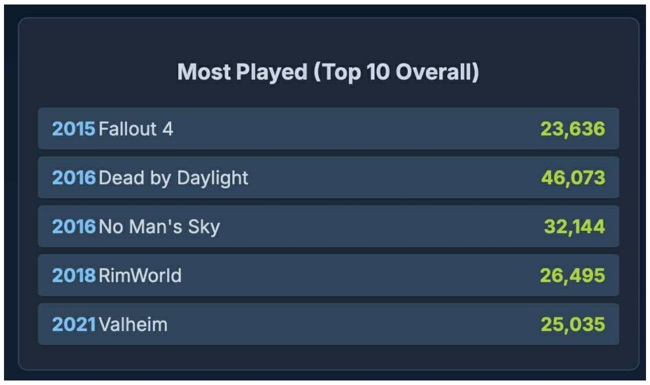
- `/api/most-played?year=2019`: Verified only 2019 games were returned and sorted by TotalPeakUsers.
- `/api/sales-revenue`: SUM(Launch_Price * Estimated_Owners) for Linux games matched API output.
- `/api/trending-games`: Correct aggregation of GamesReleased across all months for a given year.

6.3.3 Results. All API endpoints responded successfully with correct JSON structures. Parameterized filtering operated as expected. Aggregations (SUM, COUNT, AVG), groupings (GROUP BY), ordering (ORDER BY), and summaries (WITH ROLLUP) matched manual SQL verification, confirming accurate OLAP operations.

6.4 Performance Testing

6.4.1 Evaluation Procedure and Measurement. Query execution times were measured using MySQL `EXPLAIN ANALYZE`. Average execution time over multiple runs was computed for each analytical query.

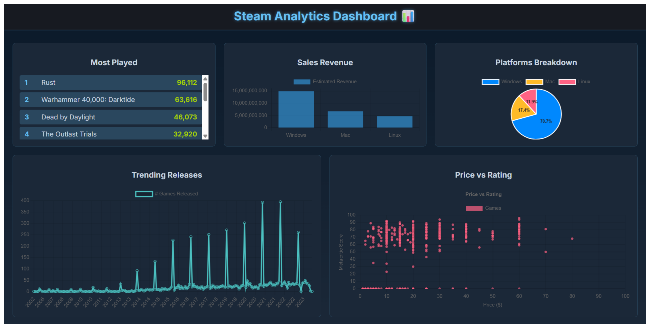
Queries executed efficiently, ranging from 0.000 to 0.047 seconds. Indexes on primary keys, foreign keys, and frequently filtered columns (e.g., Peak_CCU) were effectively utilized. Normalized



Most Played (Top 10 Overall)

2015	Fallout 4	23,636
2016	Dead by Daylight	46,073
2016	No Man's Sky	32,144
2018	RimWorld	26,495
2021	Valheim	25,035

Figure 14: OLAP Functional Testing: Most Played and Trending Games API outputs.



Steam Analytics Dashboard

Most Played

1	Rust	96,112
2	Warhammer 40,000: Darktide	63,616
3	Dead by Daylight	46,073
4	The Outlast Trials	32,920

Figure 15: OLAP Functional Testing: Sales Revenue and Platform Breakdown API outputs.

Table 1: OLAP Query Performance

Query (short)	Time (sec)
1. Peak CCU/playtime by year-platform	0.047
2. Game count & total CCU by timeline	0.047
3. Revenue by platform	0.031
4. Price vs Metacritic (sample 500)	0.000
5. Games/year by platform (rollup)	0.015

schema design, proper indexing, and correct numeric data types contributed to optimized OLAP performance.

6.4.2 *Factors Affecting Performance.*

- **Database Structure:** Highly normalized schema ensures data integrity but increases JOIN complexity.
- **Indexing:** Indexes improved query performance by enabling efficient lookups. Over-indexing was avoided to prevent unnecessary storage and update overhead.
- **Query Complexity:** Aggregations, rollups, and multi-table joins inherently add computation time, mitigated by careful design and indexing.
- **Hardware:** No specialized hardware optimization; tests were conducted on standard development machines.

6.4.3 Analysis and Discussion. The results demonstrate that the ETL process and OLAP queries are robust, accurate, and performant. Key insights include:

- ETL transformations maintained data accuracy and integrity, supporting reliable analytics.
- Parameter-driven OLAP queries enable dynamic analysis at multiple granularity levels (roll-up, drill-down, slice, dice).
- Indexing and schema design significantly enhanced query performance to validate the design decisions made in the dimensional model.

7 Conclusion

In this project, we consolidated three Steam games datasets (CSV, JSON, XLSX) into a centralized data warehouse and implemented an ETL process to ensure efficient, accurate, and up-to-date data loading. On this foundation, we developed the Steam Games Analytics Dashboard, featuring five interactive OLAP reports with roll-up, drill-down, slice, dice, and pivot operations, supported by optimized queries for faster analytics.

This project highlighted the importance of a data warehouse for integrating diverse sources, preserving history, and enabling consistent decision-making. The ETL process ensures the warehouse remains current, while OLAP enables analysis of historical and aggregated data, complementing OLTP systems. Query optimization, including indexing, caching, and schema design, is essential to maintain performance and scalability.

Our dashboard provides end users with actionable insights into game trends, sales, and engagement. For database developers, this project demonstrates best practices in data integration, dimensional modeling, ETL implementation, and query optimization, illustrating how thoughtful design transforms raw data into meaningful, timely insights.

8 References

[1] Wikipedia. 2025. Steam (service). https://en.wikipedia.org/wiki/Steam_(service). Accessed: Oct 24, 2025.
[2] Wikipedia. 2025. Data warehouse. https://en.wikipedia.org/wiki/Data_warehouse. Accessed: Oct 24, 2025.
[3] IBM. 2021. What is ETL (Extract, Transform, Load)? IBM Think. https://www.ibm.com/think/topics/etl. Accessed: Oct 24, 2025.
[4] Amazon Web Services (AWS). 2024. What is OLAP? – Online Analytical Processing Explained. https://aws.amazon.com/what-is/olap/. Accessed: Oct 24, 2025.
[5] GeeksforGeeks. 2025. Query Processing in SQL. https://www.geeksforgeeks.org/sql/sql-query-processing/. Accessed: Oct 24, 2025.
[6] S. Chaudhuri. 1998. An overview of query optimization in relational systems. Stanford CS345d. https://web.stanford.edu/class/cs345d-01/rl/chaudhuri98.pdf. Accessed: Oct 24, 2025.
[7] E. F. Codd. 1970. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6), 377–387.
[8] MySQL. MySQL 8.0 Reference Manual - Chapter 8 Optimization. https://dev.mysql.com/doc/refman/8.0/en/optimization.html.

9 Declarations

9.1 Declaration of Generative AI in Scientific Writing

During the preparation of this work the authors used ChatGPT to assist with the following tasks: Grammar Correction and Enhancement, Information Validation, and LaTeX Use and Formatting.

After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

9.2. Record of Contribution

Abergos, Mark Gabriel: Software Development: OLAP App. Technical Report: Results and Analysis.

Boado, Jorem Ivan: Software Development: Data Warehouse (Conceptualized & Revised Star Schema Design), ETL Script. Technical Report: ETL Script, Conclusion, Results & Analysis.

Celemen, Mark Edrian: Software Development: Data Warehouse (Prepared Datasets, Conceptualized Dimensional & Fact Table and Star Schema Design), OLAP App (Prepared Reports and Visualizations, Conceptualized Dashboard Design). Technical Report: Introduction, References, Declaration, ACM Publication Format.

Kyle Elijah Gan: Software Development: OLAP App. Technical Report: None.

Bryant Kimm Gonzales: Software Development: Data Warehouse (Star Schema Design Conceptualization & Revisions), ETL Script. Technical Report: Data Warehouse, Query Optimization.

Audrey Lauren Stinson: Software Development: Data Warehouse (Star Schema Design Conceptualization & Revisions), ETL Script, OLAP App (Reports and Visualizations Fixes). Technical Report: ETL, Results & Analysis, Query Optimization.

John Vincent Torres: Software Development: OLAP App. Technical Report: OLAP Application.