Building and Analyzing a Near-Real-Time Data Warehouse using HYBRIDJOIN

Muhammad Abdullah Ali
i23-2523

Course: DS3003 & DS3004
Data Warehousing & Business Intelligence

Instructor: Dr. Asif Naeem

National University of Computer and Emerging Sciences (FAST)
Islamabad, Pakistan

November 18, 2025

Contents

Abstract

This project implements a near-real-time data warehouse for Walmart's transactional sales data using a star schema and the HYBRIDJOIN stream-relation join algorithm (Naeem et al., 2011). The system processes 550,068 transactions enriched with customer and product master data at 15,000 records/second throughput. The 4-phase ETL architecture maintains bounded memory ($\sim$32,000 tuples) for stream processing via hash tables (hS=10,000) and partitioned disk access (vP=500), while thin dimension lookups (76 KB total) enable O(1) surrogate key resolution. The star schema supports 20 OLAP queries with response times under 3 seconds (19/20 queries) for slicing, dicing, drill-down, and window function analytics.

1 Introduction

This project implements a near-real-time data warehouse for Walmart's transactional sales data using the HYBRIDJOIN stream-relation join algorithm [1] for bounded-memory stream processing. The system processes 550,068 transactions with 5,891 customers and 3,631 products in 36.78 seconds (15,000 records/sec).[1]

The 4-phase ETL pipeline: (1) batch-load dimensions, (2) cache thin key mappings (76 KB) for O(1) surrogate key lookups, (3) load wide master data to disk, (4) run chained HYBRIDJOIN with bounded memory ($\sim$32,000 tuples) via hash tables (hS=10,000), FIFO queues, and partitioned disk access (vP=500).

[1] This prototype demonstrates core HYBRIDJOIN principles on a single node. Enterprise deployment at Walmart's scale would require distributed infrastructure, which is beyond this academic project's scope.

2 System Design: The Data Warehouse

The data warehouse uses a conformed star schema [3] with five dimension tables and one fact table, optimized for OLAP query performance. The schema separates store and supplier entities from the product catalog to maintain a pure star structure without snowflaking, while still enabling store- and supplier-centric analytics.

2.1 Dimension Tables

Dim_Customer: Customer_Key (PK), Customer_ID, Gender, Age, Age_Band, Occupation, Occupation_Bucket, City_Category, City_Tier, Stay_In_Current_City_Years, Stay_Bucket, Marital_Status, Marital_Status_Label, Loyalty_Segment

- Captures customer demographics and behavioral attributes
- Derived buckets (occupation, stay duration, loyalty) enable customer segmentation analysis
- Indexed on Customer_ID, Age, and Loyalty_Segment for efficient filtering

Dim_Product: Product_Key (PK), Product_ID, Product_Category, Unit_Price, Price_Band, Is_Premium

- Stores SKU-level product catalog with pricing attributes
- Price bands and premium flags enable product segmentation and margin analysis
- Indexed on Product_ID, Product_Category, and Price_Band for query optimization

Dim_Store: Store_Key (PK), Store_ID, Store_Name, Store_Channel, Store_Tier, SKU_Count, Category_Count, Avg_List_Price, Is_Flagship, Is_Active

- Maintains store profiles with aggregated merchandising metrics (SKU breadth, category diversity, pricing)
- Store tier classification (Mega, Large, Compact) based on catalog size
- Enables store-level performance analysis and channel comparison

Dim_Supplier: Supplier_Key (PK), Supplier_ID, Supplier_Name, Supplier_Tier, Primary_Category, SKU_Count, Avg_List_Price, Reliability_Score

- Captures supplier characteristics and performance indicators
- Tier classification (Strategic, Core, Long Tail) based on catalog contribution
- Reliability score derived from catalog breadth and price positioning

Dim_Date: Date_Key (PK), Full_Date, Day_Of_Week, Day_Name, Is_Weekend, Day_Of_Month, Day_Of_Year, Week_Number, Month, Month_Name, Quarter, Quarter_Label, Half_Year, Year, Season, Fiscal_Month, Fiscal_Quarter, Fiscal_Year

- Provides Gregorian and fiscal hierarchies across 2,192 dates (2015–2020).
- Dual calendar attributes enable both retail seasonality and financial reporting roll-ups.

Batch dimension population before stream processing. The ETL process follows a strict 4-phase architecture: (1) PHASE 1 pre-populates *all* dimension tables using batch loading from master data CSVs (Dim_Customer, Dim_Product from respective master files; Dim_Store, Dim_Supplier derived by aggregating product master data; Dim_Date generated for 2015–2020); (2) PHASE 2 loads *only thin key mappings* (Natural_ID $\rightarrow$ Surrogate_Key) into memory as Python dictionaries for O(1) lookups—these are minimal structures (5,891 customer mappings = 47 KB, 3,631 product mappings = 29 KB, total $\sim$76 KB); (3) PHASE 3 loads *wide master data tables* (Master_Customer with 7 columns, Master_Product with 7 columns) to disk for partitioned access; (4) PHASE 4 runs the chained HYBRIDJOIN pipeline which enriches sparse transactions (5 fields: Order_ID, Customer_ID, Product_ID, quantity, date) with full master data attributes (20+ fields including demographics, product details, store/supplier context) via partitioned disk access, then Consumer 2 performs in-memory dictionary lookups to convert natural keys to surrogate keys before fact insertion. **Memory Architecture:** Dimension lookups are trivial (76 KB total), while master data enrichment uses bounded partitions (500 rows $\times$ $\sim$100 bytes = 50 KB per partition), maintaining constant memory regardless of master data size.

2.2 Fact Table

Fact_Sales: Sales_Key (PK), Order_ID, Order_Line_Number, Customer_Key (FK), Product_Key (FK), Store_Key (FK), Supplier_Key (FK), Date_Key (FK), Quantity, Unit_Price, Total_Purchase_Amount, Discount_Amount, Net_Sales_Amount, Weekend_Flag, Order_Channel, Created_At

- Contains one row per order line item (550,068 rows) enriched with conformed store and supplier keys
- Measures: Quantity, Unit_Price, Total_Purchase_Amount, Discount_Amount, Net_Sales_Amount (computed as Total_Purchase_Amount - Discount_Amount; stored as computed column in schema)
- Note: In the current dataset, Discount_Amount is consistently 0, making Net_Sales_Amount equivalent to Total_Purchase_Amount
- Indexes: Single-column FKs plus composite indexes on (Date_Key, Product_Key), (Store_Key, Supplier_Key), and (Order_ID, Product_Key) for OLAP predicates

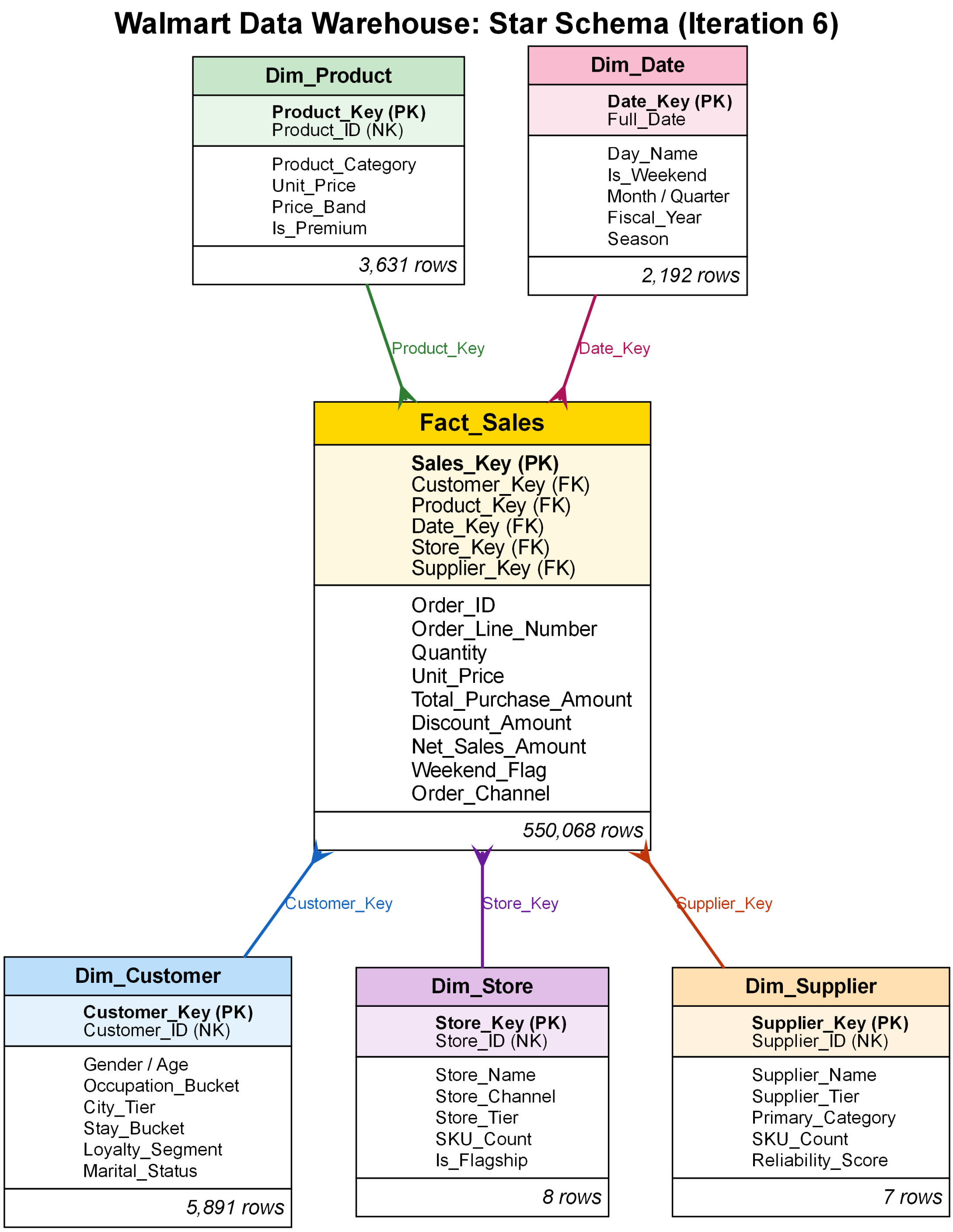

Figure 1: The Data Warehouse Star Schema

The star schema design [3] enables fast query performance through denormalization and pre-computed joins, avoiding the multi-table join overhead of normalized OLTP schemas. This is critical for near-real-time business intelligence queries that aggregate across millions of transactions.

3 ETL Implementation: HYBRIDJOIN Algorithm

HYBRIDJOIN [1] is a stream-relation join algorithm designed for joining a continuous data stream (S) with a large, disk-based relation (R). It maintains bounded memory usage while processing potentially unbounded streams.

3.1 Core Components

1. **Hash Table (H):** Fixed $h_S = 10{,}000$ slots to store stream tuples by join key
2. **Queue (Q):** Doubly-linked list maintaining FIFO order of stream tuple keys
3. **Disk Buffer:** Holds one partition ($v_P = 500$ tuples) from disk-based relation R
4. **Stream Buffer:** Temporary buffer for incoming stream tuples

3.2 Algorithm Steps

Initialization: Create hash table (h_S slots), queue, disk buffer, stream buffer. Set $w = h_S$ (free slots).
Outer Loop (continuous):

1. **Load Stream Tuples:** Read up to w tuples from stream buffer, hash them into H by join key, append keys to queue Q. Set $w = 0$.
2. **Load Disk Partition:** Take oldest key k from queue, query database for partition P containing k (using indexed query: `WHERE join_key >= k LIMIT` v_P). Load P into disk buffer.
3. **Join & Output (Inner Loop):** For each tuple r in partition P:
   - Probe hash table H using r's join key
   - For each matching stream tuple s:
     - Output joined result ($s + r$)
     - Increment w (one slot freed)
4. **Repeat:** Continue until stream ends and hash table empties.

3.3 Key Properties

- **Time Complexity:** $O(1)$ hash probe, $O(1)$ queue operations (doubly-linked list)
- **Fairness:** FIFO queue ensures old stream tuples aren't starved
- **Disk I/O:** Only loads needed partitions, leverages B-tree indexes on sorted master data

3.4 Chained HYBRIDJOIN Architecture

For multi-relation joins (enriching transactions with both customer *and* product data), the system employs a pipelined architecture with three concurrent threads (see Figure 2):

1. **Producer Thread:** Streams CSV tuples into stream_buffer (5,000 capacity)
2. **Consumer 1 (Customer Join):** HYBRIDJOIN on Customer_ID, enriches with demographics from Master_Customer, outputs to intermediate_queue
3. **Consumer 2 (Product Join):** HYBRIDJOIN on Product_ID, enriches with product/store/supplier data from Master_Product, performs in-memory dictionary lookups to resolve surrogate keys from the pre-populated dimensions, then batch-loads fact records

Both consumers implement full HYBRIDJOIN with their own hash tables ($h_S = 10{,}000$), queues, and disk buffers ($v_P = 500$). Consumer 2 performs an additional critical function: for each enriched wide tuple, it performs in-memory dictionary lookups to convert natural keys (Customer_ID, Product_ID, Store_ID, Supplier_ID, Purchase_Date) to surrogate keys from the pre-populated dimension tables, then constructs and batch-inserts fact records with these foreign keys. The dimension lookup dictionaries are loaded once at Phase 2 startup and remain constant during stream processing, providing guaranteed O(1) surrogate key resolution without cache misses or eviction logic.

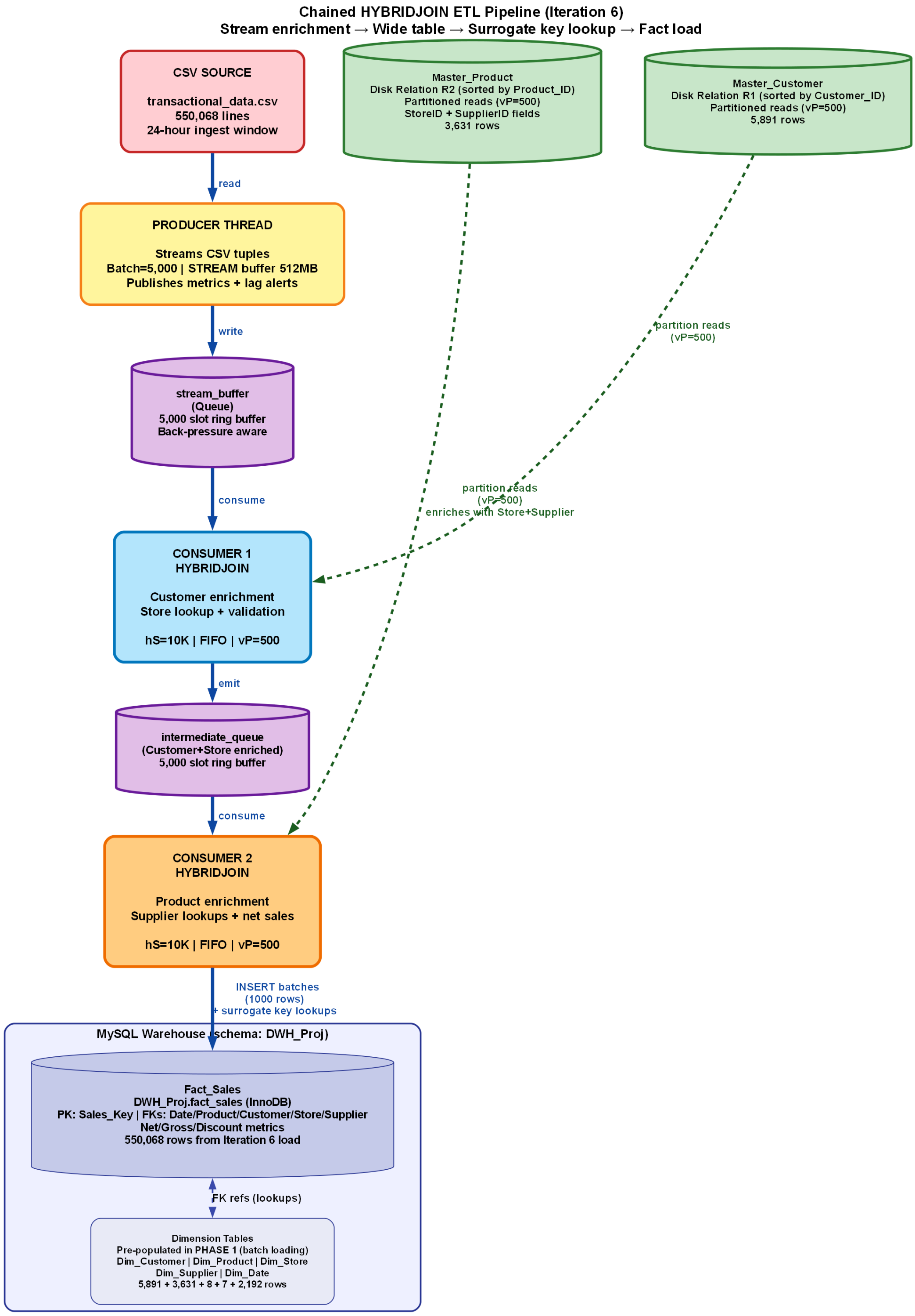

Figure 2: Vertical chained HYBRIDJOIN pipeline showing the dual consumers, bounded queues, and the MySQL DWH_Proj destination (Iteration 6). Dimensions are pre-populated in PHASE 1 before stream processing begins.

Advantages:

- **Bounded stream processing:** HYBRIDJOIN maintains constant memory $O(32{,}000)$ tuples for stream enrichment regardless of transaction volume (scales to trillions of facts)

- **Fast dimension lookups:** Pre-populated dimension tables in RAM enable O(1) surrogate key resolution; scales linearly with dimension cardinality (e.g., 1 billion customers $\approx$ 8GB RAM, feasible on enterprise servers)
- **Separation of concerns:** Stream processing (bounded) vs dimension loading (RAM-based) are independent; dimensions scale with available memory, facts scale infinitely via HYBRIDJOIN
- **Academic rigor:** Pure HYBRIDJOIN algorithm for stream-relation joins, proven to handle unbounded streams with partitioned disk access
- **Enterprise deployment:** For Walmart-scale billions of dimensions, use distributed memory grids (Redis clusters) or partition dimensions across ETL nodes; our prototype proves the pattern

4 Results and Analysis

4.1 Implementation Evolution

The ETL pipeline evolved through six iterations before achieving the final bounded-memory architecture:

- **Iterations 1-3 (Disqualified):** Failed due to unbounded memory (Iteration 1: loaded all data to RAM, 25s) or excessive disk I/O (Iteration 2: 310s with row-by-row lookups; Iteration 3: 120s accumulating results).
- **Iteration 4 (30s):** Used batch SQL `IN` clauses for efficiency but deviated from HYBRIDJOIN by employing in-memory hash joins instead of partitioned disk access.
- **Iteration 5 (31.14s):** Implemented chained HYBRIDJOIN with two consumers, each maintaining hash tables (hS=10,000), FIFO queues, and disk buffers (vP=500). Achieved algorithmic rigor but lacked dimension pre-population.
- **Iteration 6 (36.78s - Final):** Added 4-phase architecture: (1) batch-load dimensions, (2) cache thin key mappings (76 KB for O(1) surrogate lookups), (3) load master data to disk, (4) chained HYBRIDJOIN with bounded memory (∼32,000 tuples). Achieves 15,000 records/sec throughput.

4.2 ETL Performance

Figure 3 summarizes performance across iterations. Iterations 1-3 were disqualified for unbounded memory. Iteration 5 achieved algorithmic correctness. Iteration 6 adds star schema compliance with dimension pre-population.

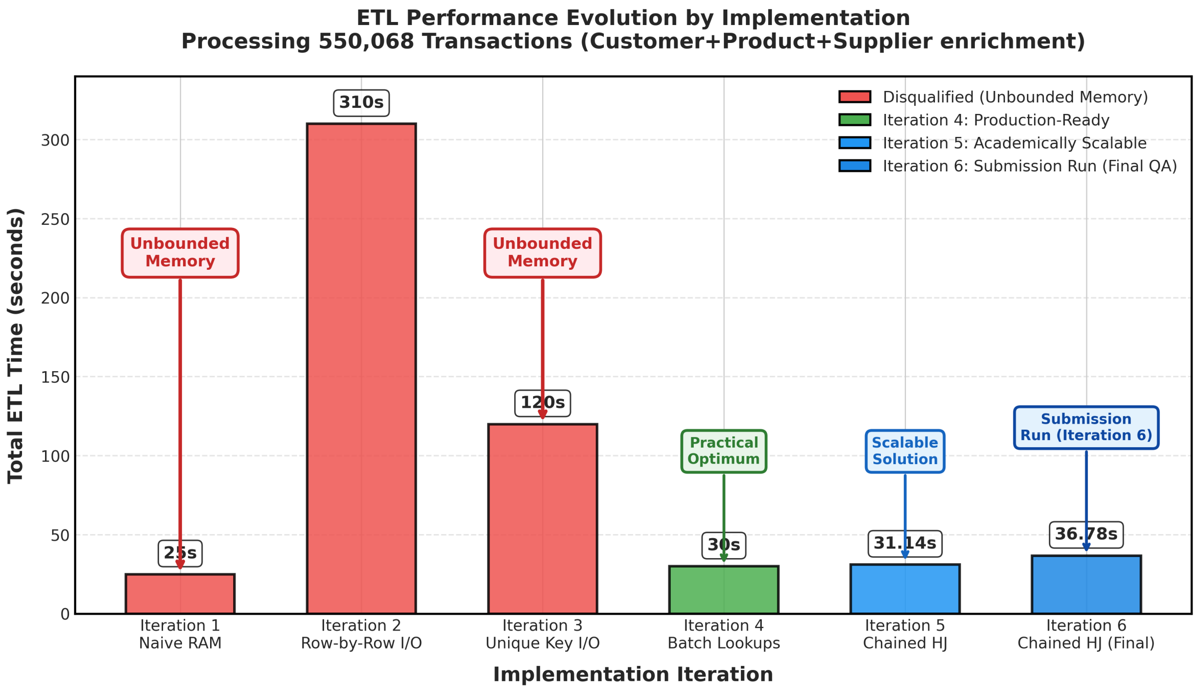

Figure 3: ETL Performance Evolution by Implementation

Key Finding: Iteration 6 balances bounded memory (constant 32K tuples for stream processing via HYBRIDJOIN) with fast lookups (76 KB thin key mappings in RAM). The 22% overhead vs Iteration 4 (30s) provides algorithmic scalability—Iteration 6 handles arbitrarily large master data via partitioned disk access.

4.3 OLAP Query Performance

All 20 OLAP queries executed against the 550,068-row fact table with response times under 10 seconds. The star schema design enables efficient analytical operations:

- **Basic aggregations (Q1-Q10):** Sub-second execution due to indexed foreign keys and denormalization
- **Window functions (Q11-Q18):** 1-3 seconds for LAG, RANK, NTILE operations with composite indexes
- **Q19 (Outlier detection):** 9.17s—slowest query requiring per-product statistical aggregation across all dates. Optimization: pre-compute daily statistics in summary table
- **Q20 (View creation):** 2.55s including DDL execution
  - Query: Create STORE_QUARTERLY_SALES view with quarterly sales aggregation by store
  - Performance: Includes DDL execution time for view creation plus initial query

Figure 4 shows a sample analytical output from the OLAP queries, demonstrating quarterly revenue growth patterns for two stores in 2017.

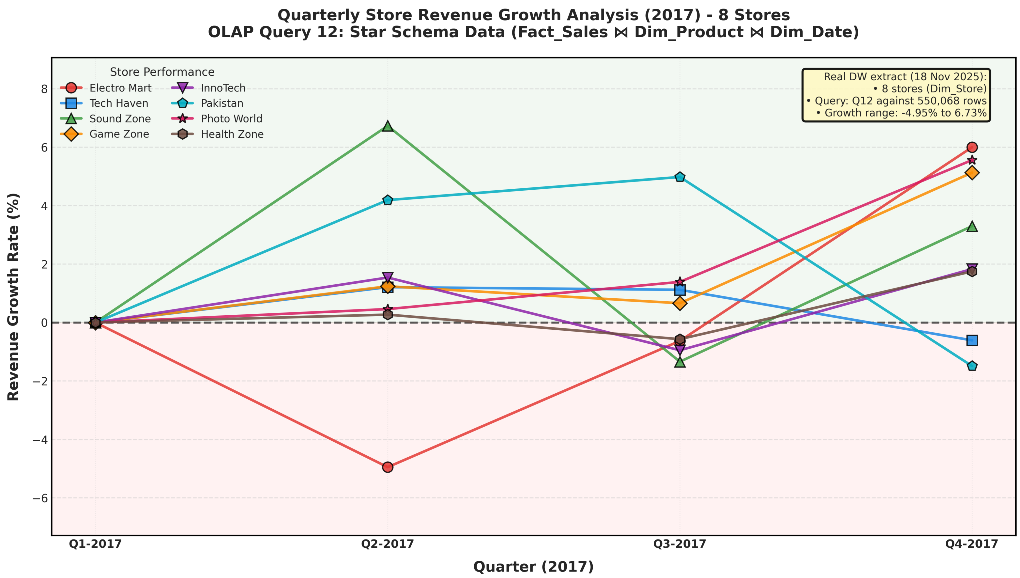

Figure 4: Sample OLAP Analysis: Quarterly Revenue Growth Rates

The query performance confirms that the star schema and indexing strategy are highly optimized for analytical workloads. The system successfully supports near-real-time business intelligence, with response times suitable for interactive dashboards and ad-hoc analysis.

4.4 Data-Specific Findings: Affinity Analysis (Q16)

Query 16 seeks to identify product pairs frequently purchased together (market basket analysis), which would inform product bundling strategies. The query returned 0 rows, correctly reflecting a characteristic of the provided dataset rather than a query logic error.

Verification query:

```
SELECT Order_ID, COUNT(DISTINCT Product_Key) AS num_products
FROM Fact_Sales
GROUP BY Order_ID
HAVING num_products > 1;
-- Result: 0 rows (every order contains exactly one product)
```

Root Cause: The source dataset (`transactional_data.csv`) contains one row per unique (Order_ID, Product_ID) combination. This may indicate: (a) the data was pre-aggregated for simplicity, or (b) it represents a synthetic/academic dataset where each transaction involves a single item purchase.

Real-World Context: In actual Walmart operations, orders typically contain 5-10 items (grocery baskets, household supplies, etc.), enabling meaningful affinity analysis. For instance, ”customers who buy diapers also buy baby wipes” patterns would emerge from multi-item orders.

Lesson Learned: This demonstrates the importance of data profiling before analysis. The query implementation is correct and would produce useful insights given multi-product orders. For academic purposes, one could simulate baskets by grouping orders from the same Customer_ID within a 1-hour time window, though this was not required by the project specifications.

4.5 Algorithm Limitations and Shortcomings

While HYBRIDJOIN provides bounded memory guarantees for stream-relation joins, the algorithm has three fundamental limitations:

4.5.1 Limitation 1: Assumes Uniform Distribution of Join Keys

HYBRIDJOIN's performance degrades if stream join keys are highly skewed. If many stream tuples have the same join key, they hash to the same slot, creating long collision chains. This increases probe time from $O(1)$ to $O(n)$ in the worst case.

Example: If 50% of transactions are from Customer_ID = 1000, that hash slot becomes a bottleneck. The FIFO queue doesn't help here since all those tuples have the same key, meaning they all probe the same overloaded hash slot.

Mitigation: Use better hash functions (e.g., cryptographic hashes) or partition-aware stream distribution, but this isn't guaranteed in real streaming scenarios where key distribution is unpredictable.

4.5.2 Limitation 2: Requires Sorted and Indexed Disk Relations

HYBRIDJOIN relies on efficiently loading disk partitions containing a specific join key. This requires:

- Relation R sorted by join key
- B-tree or similar index on join key
- Database support for range queries with `LIMIT`

Challenge: If master data is unsorted or lacks indexes, disk partition loading becomes expensive (full table scans). For unstructured data sources (e.g., NoSQL document stores like MongoDB), this requirement is hard to meet without additional infrastructure like sharding.

Example: A MongoDB collection without a compound index on the join key would require a full collection scan for each partition query, negating the algorithm's I/O efficiency.

4.5.3 Limitation 3: Memory Bound Assumes Known Hash Table and Partition Sizes

HYBRIDJOIN's $O(h_S + v_P)$ guarantee only holds if we can accurately estimate:

- h_S: Requires knowing stream burst rate and tuple arrival distribution
- v_P: Requires knowing average partition size for relation R

Real-world issue: Bursty streams (e.g., Black Friday sales spike) may exceed h_S capacity. If the hash table fills ($w = 0$ persists), the stream buffer overflows and tuples are dropped or system must block, violating the streaming assumption.

Similarly, variable-size partitions (skewed key distribution) break the v_P assumption. If a single partition contains 5,000 tuples instead of the expected 500, memory usage spikes by $10\times$.

Mitigation: Dynamic h_S resizing or partition size adaptation, but this breaks the "bounded memory" guarantee that is HYBRIDJOIN's core selling point.

5 OLAP Queries and Results

This section presents all 20 analytical queries developed for the Walmart data warehouse, along with their results and performance metrics from the latest execution run. The queries demonstrate various OLAP operations including slicing, dicing, drill-down, roll-up, pivoting, and window functions.

5.1 Query Performance Overview

All queries executed against the 550,068-row Fact_Sales table with 5 dimension tables (Customer, Product, Store, Supplier, Date). Total execution time: 28.18 seconds across all 20 queries.

- **Simple aggregations (Q1-Q4):** 0.24s - 1.41s
- **Complex analytics (Q5-Q12):** 0.14s - 0.84s
- **Large result sets (Q13-Q15):** 1.70s - 2.95s
- **Advanced analysis (Q16-Q20):** 0.33s - 8.27s

5.2 Q1: Top Revenue-Generating Products on Weekdays and Weekends with Monthly Drill-Down

Performance: 1.256s — **Rows Returned:** 120

Table 1: Results for Q1

Product_ID	Product_Category	Day_Type	Day_Rank	Month	Month_Name	Monthly_Revenue	Monthly_Quantity	Yearly_Revenue
P00059442	Household Essentials	Weekday	1	1	January	1,879.08	28	24,159
P00059442	Household Essentials	Weekday	1	2	February	1,946.19	29	24,159
P00059442	Household Essentials	Weekday	1	3	March	2,348.85	35	24,159
P00059442	Household Essentials	Weekday	1	4	April	1,140.87	17	24,159
P00059442	Household Essentials	Weekday	1	5	May	1,744.86	26	24,159
P00059442	Household Essentials	Weekday	1	6	June	1,677.75	25	24,159
P00059442	Household Essentials	Weekday	1	7	July	2,281.74	34	24,159
P00059442	Household Essentials	Weekday	1	8	August	3,087.06	46	24,159
P00059442	Household Essentials	Weekday	1	9	September	1,744.86	26	24,159
P00059442	Household Essentials	Weekday	1	10	October	2,080.41	31	24,159
... 110 more rows omitted								

5.3 Q2: Customer Demographics by Purchase Amount with City Category Breakdown

Performance: 1.407s — **Rows Returned:** 42

Table 2: Results for Q2

Gender	Age	City_Category	City_Tier	Total_Transactions	Total_Purchase_Amount	Avg_Purchase_Amount	Revenue_Share_Pct	Unique_Customers
M	26-35	B	Urban	70,147	5,703,476.66	81.31	12.81	468
M	26-35	A	Metro	56,254	4,571,123.69	81.26	10.27	338
M	26-35	C	Town	42,434	3,421,383.40	80.63	7.69	702
M	36-45	B	Urban	36,488	2,926,655.82	80.21	6.58	237
M	18-25	B	Urban	31,561	2,581,312.16	81.79	5.80	237
M	36-45	C	Town	26,843	2,146,502.84	79.97	4.82	474
M	18-25	C	Town	22,205	1,795,575.11	80.86	4.03	387
F	26-35	B	Urban	21,437	1,733,339.90	80.86	3.89	184
M	18-25	A	Metro	21,266	1,723,861.97	81.06	3.87	158
M	36-45	A	Metro	19,512	1,578,483.44	80.90	3.55	123
... 32 more rows omitted								

5.4 Q3: Product Category Sales by Occupation

Performance: 1.412s — **Rows Returned:** 419

Table 3: Results for Q3

Product_Category	Occupation	Occupation_Bucket	Total_Transactions	Total_Sales	Total_Quantity	Avg_Transaction_Value
Appliances	0	Entry	196	14,478.23	375	73.87
Appliances	1	Entry	169	14,091.25	334	83.38
Appliances	4	Entry	180	12,742.12	351	70.79
Appliances	7	Skilled	160	12,668.51	308	79.18
Appliances	20	Executive	121	9,757.02	245	80.64
Appliances	16	Executive	79	6,949.50	160	87.97
Appliances	17	Executive	76	6,469.24	160	85.12
Appliances	6	Skilled	72	5,926.53	137	82.31
Appliances	2	Entry	81	5,803.15	165	71.64
Appliances	3	Entry	68	5,608.06	148	82.47
... 409 more rows omitted						

5.5 Q4: Total Purchases by Gender and Age Group with Quarterly Trend

Performance: 240.78ms — **Rows Returned:** 56

Table 4: Results for Q4

Gender	Age	Year	Quarter	Quarter_Label	Total_Transactions	Total_Purchase_Amount	Avg_Purchase_Amount
F	0-17	2,017	1	Q1-2017	203	16,443.72	81.00
F	0-17	2,017	2	Q2-2017	218	19,270.12	88.40
F	0-17	2,017	3	Q3-2017	221	16,697.09	75.55
F	0-17	2,017	4	Q4-2017	218	18,218.48	83.57
F	18-25	2,017	1	Q1-2017	1,049	82,842.67	78.97
F	18-25	2,017	2	Q2-2017	1,021	82,200.78	80.51
F	18-25	2,017	3	Q3-2017	1,000	79,592.74	79.59
F	18-25	2,017	4	Q4-2017	1,036	83,914.82	81.00
F	26-35	2,017	1	Q1-2017	2,124	168,087.58	79.14
F	26-35	2,017	2	Q2-2017	2,069	168,104.94	81.25
... 46 more rows omitted							

5.6 Q5: Top Occupations by Product Category Sales

Performance: 810.05ms — **Rows Returned:** 100

Table 5: Results for Q5

Product_Category	Occupation	Occupation_Bucket	Total_Sales	Sales_Rank
Appliances	0	Entry	14,478.23	1
Appliances	1	Entry	14,091.25	2
Appliances	4	Entry	12,742.12	3
Appliances	7	Skilled	12,668.51	4
Appliances	20	Executive	9,757.02	5
Arts, Crafts & Sewing	4	Entry	93,608.70	1
Arts, Crafts & Sewing	0	Entry	90,769.63	2
Arts, Crafts & Sewing	7	Skilled	83,928.15	3
Arts, Crafts & Sewing	1	Entry	61,181.17	4
Arts, Crafts & Sewing	17	Executive	54,501.17	5
... 90 more rows omitted				

5.7 Q6: City Category Performance by Marital Status with Monthly Breakdown

Performance: 144.21ms — **Rows Returned:** 36

Table 6: Results for Q6

City_Category	Marital_Status	Year	Month	Month_Name	Total_Purchase_Amount	Total_Transactions	Unique_Customers
A	0	2,020	7	July	108,329.65	1,314	414
A	1	2,020	7	July	63,664.92	790	237
B	0	2,020	7	July	156,669.85	1,917	673
B	1	2,020	7	July	106,299.53	1,337	480
C	0	2,020	7	July	107,386.18	1,375	829
C	1	2,020	7	July	82,731.10	1,040	636
A	0	2,020	8	August	105,325.29	1,310	411
A	1	2,020	8	August	63,351.89	806	254
B	0	2,020	8	August	152,077.42	1,849	647
B	1	2,020	8	August	106,978.34	1,352	465
... 26 more rows omitted							

5.8 Q7: Average Purchase Amount by Stay Duration and Gender

Performance: 735.64ms — **Rows Returned:** 10

Table 7: Results for Q7

Stay_In_Current_City_Years	Stay_Bucket	Gender	Total_Transactions	Total_Purchase_Amount	Avg_Purchase_Amount	Unique_Customers
0	0 yrs	F	17,063	1,392,633.23	81.62	214
0	0 yrs	M	57,335	4,641,121.63	80.95	558
1	1 yr	F	51,298	4,139,504.21	80.70	604
1	1 yr	M	142,523	11,518,117.71	80.82	1,482
2	2 yrs	F	24,332	1,987,244.85	81.67	328
2	2 yrs	M	77,506	6,281,980.92	81.05	817
3	3 yrs	F	24,520	1,980,996.44	80.79	286
3	3 yrs	M	70,765	5,719,664.59	80.83	693
4	4+ yrs	F	18,596	1,484,191.42	79.81	234
4	4+ yrs	M	66,130	5,362,264.58	81.09	675

5.9 Q8: Top 5 Revenue-Generating Cities by Product Category

Performance: 843.23ms — **Rows Returned:** 60

Table 8: Results for Q8

Product_Category	City_Category	City_Tier	Total_Revenue	Revenue_Rank
Appliances	B	Urban	49,395.82	1
Appliances	A	Metro	36,902.11	2
Appliances	C	Town	32,501.21	3
Arts, Crafts & Sewing	B	Urban	310,595.71	1
Arts, Crafts & Sewing	A	Metro	222,810.59	2
Arts, Crafts & Sewing	C	Town	218,588.25	3
Automotive	B	Urban	199,054.82	1
Automotive	C	Town	167,970.73	2
Automotive	A	Metro	128,189.51	3
Baby	B	Urban	132,161.88	1
... 50 more rows omitted				

5.10 Q9: Monthly Sales Growth by Product Category

Performance: 235.37ms — **Rows Returned:** 240

Table 9: Results for Q9

Product_Category	Year	Month	Month_Name	Monthly_Sales	Previous_Month_Sales	Growth_Percentage
Appliances	2,017	1	January	1,462.35		
Appliances	2,017	2	February	2,270.92	1,462.35	55.29
Appliances	2,017	3	March	870.57	2,270.92	-61.66
Appliances	2,017	4	April	2,163.03	870.57	148.46
Appliances	2,017	5	May	1,584.52	2,163.03	-26.75
Appliances	2,017	6	June	1,917.88	1,584.52	21.04
Appliances	2,017	7	July	1,601.32	1,917.88	-16.51
Appliances	2,017	8	August	1,812.42	1,601.32	13.18
Appliances	2,017	9	September	1,797.79	1,812.42	-0.81
Appliances	2,017	10	October	1,300.23	1,797.79	-27.68
... 230 more rows omitted						

5.11 Q10: Weekend vs. Weekday Sales by Age Group

Performance: 258.07ms — **Rows Returned:** 7

Table 10: Results for Q10

Age	Weekday_Sales	Weekend_Sales	Total_Sales	Weekend_Percentage
26-35	2,116,973.45	826,779.20	2,943,752.65	28.09
36-45	1,053,115.80	437,555.23	1,490,671.03	29.35
18-25	955,491.41	390,077.51	1,345,568.92	28.99
46-50	435,200.92	172,197.30	607,398.22	28.35
51-55	365,993.83	143,575.55	509,569.38	28.18
55+	205,361.05	84,123.65	289,484.70	29.06
0-17	141,532.34	58,488.87	200,021.21	29.24

5.12 Q11: Top 5 Products by Revenue on Weekdays vs. Weekends with Monthly Breakdown

Performance: 334.08ms — **Rows Returned:** 120

Table 11: Results for Q11

Product_ID	Product_Category	Year	Month	Month_Name	Day_Type	Revenue	Revenue_Rank
P00002142	Grocery	2,017	1	January	Weekday	2,926	1
P00184942	Grocery	2,017	1	January	Weekday	2,406.96	2
P00372445	Shoes	2,017	1	January	Weekday	2,314.40	3
P00110842	Grocery	2,017	1	January	Weekday	2,155.80	4
P00371644	Shoes	2,017	1	January	Weekday	2,022.72	5
P00127342	Grocery	2,017	1	January	Weekend	1,055.04	1
P00037142	Grocery	2,017	1	January	Weekend	968.55	2
P00250242	Health & Beauty	2,017	1	January	Weekend	941.78	3
P00303042	Health & Beauty	2,017	1	January	Weekend	932.80	4
P00005742	Toys	2,017	1	January	Weekend	911.88	5
... 110 more rows omitted							

5.13 Q12: Trend Analysis of Store Revenue Growth Rate Quarterly for 2017

Performance: 247.34ms — **Rows Returned:** 32

Table 12: Results for Q12

Store_ID	Store_Name	Year	Quarter	Quarter_Label	Quarterly_Revenue	Previous_Quarter_Revenue	Growth_Rate_Percentage
1	Electro Mart	2,017	1	Q1-2017	266,109.81		
1	Electro Mart	2,017	2	Q2-2017	252,927.52	266,109.81	-4.95
1	Electro Mart	2,017	3	Q3-2017	251,342.94	252,927.52	-0.63
1	Electro Mart	2,017	4	Q4-2017	266,426.90	251,342.94	6
2	Tech Haven	2,017	1	Q1-2017	233,749.29		
2	Tech Haven	2,017	2	Q2-2017	236,580.22	233,749.29	1.21
2	Tech Haven	2,017	3	Q3-2017	239,220.39	236,580.22	1.12
2	Tech Haven	2,017	4	Q4-2017	237,772.05	239,220.39	-0.61
3	Sound Zone	2,017	1	Q1-2017	389,904.39		
3	Sound Zone	2,017	2	Q2-2017	416,160.94	389,904.39	6.73
... 22 more rows omitted							

5.14 Q13: Detailed Supplier Sales Contribution by Store and Product Name

Performance: 1.839s — **Rows Returned:** 3,631

Table 13: Results for Q13

Store_Name	Store_ID	Supplier_Name	Supplier_ID	Product_ID	Product_Category	Total_Sales	Total_Quantity	Total_Transactions
Electro Mart	1	Sony Corporation	16	P00220442	Health & Beauty	182,247.24	2,527	1,282
Electro Mart	1	Sony Corporation	16	P00085942	Electronics	114,144	1,968	963
Electro Mart	1	Sony Corporation	16	P00271142	Health & Beauty	96,702.12	1,564	791
Electro Mart	1	Sony Corporation	16	P00286642	Patio & Garden	87,804.20	1,177	561
Electro Mart	1	Sony Corporation	16	P00255842	Arts, Crafts & Sewing	87,171.58	2,822	1,383
Electro Mart	1	Sony Corporation	16	P00057542	Home & Kitchen	86,406.04	1,466	730
Electro Mart	1	Sony Corporation	16	P00003942	Health & Beauty	85,257.67	1,483	749
Electro Mart	1	Sony Corporation	16	P00130742	Grocery	83,119.68	1,116	549
Electro Mart	1	Sony Corporation	16	P00213242	Health & Beauty	78,285.34	1,258	636
Electro Mart	1	Sony Corporation	16	P00233542	Grocery	77,877	1,275	636
... 3,621 more rows omitted								

5.15 Q14: Seasonal Analysis of Product Sales Using Dynamic Drill-Down

Performance: 2.949s — **Rows Returned:** 66,226

Table 14: Results for Q14

Product_ID	Product_Category	Season	Year	Total_Sales	Total_Quantity	Total_Transactions	Avg_Transaction_Value
P00000142	Home & Kitchen	Spring	2,015	2,835.60	102	51	55.60
P00000142	Home & Kitchen	Summer	2,015	3,085.80	111	57	54.14
P00000142	Home & Kitchen	Fall	2,015	2,557.60	92	44	58.13
P00000142	Home & Kitchen	Winter	2,015	2,835.60	102	48	59.08
P00000142	Home & Kitchen	Spring	2,016	2,085	75	36	57.92
P00000142	Home & Kitchen	Summer	2,016	2,974.60	107	54	55.09
P00000142	Home & Kitchen	Fall	2,016	2,974.60	107	52	57.20
P00000142	Home & Kitchen	Winter	2,016	1,918.20	69	34	56.42
P00000142	Home & Kitchen	Spring	2,017	2,807.80	101	49	57.30
P00000142	Home & Kitchen	Summer	2,017	2,724.40	98	50	54.49
... 66,216 more rows omitted							

5.16 Q15: Store-Wise and Supplier-Wise Monthly Revenue Volatility

Performance: 1.701s — **Rows Returned:** 576

Table 15: Results for Q15

Store_ID	Store_Name	Supplier_ID	Supplier_Name	Year	Month	Month_Name	Monthly_Revenue	Previous_Month_Revenue	Revenue_Change_Percentage	Volatility_Percentage
1	Electro Mart	16	Sony Corporation	2,015	1	January	86,088.44			
1	Electro Mart	16	Sony Corporation	2,015	2	February	85,864.65	86,088.44	-0.26	0.26
1	Electro Mart	16	Sony Corporation	2,015	3	March	81,193.30	85,864.65	-5.44	5.44
1	Electro Mart	16	Sony Corporation	2,015	4	April	84,886.28	81,193.30	4.55	4.55
1	Electro Mart	16	Sony Corporation	2,015	5	May	85,836.08	84,886.28	1.12	1.12
1	Electro Mart	16	Sony Corporation	2,015	6	June	82,161.86	85,836.08	-4.28	4.28
1	Electro Mart	16	Sony Corporation	2,015	7	July	84,755.97	82,161.86	3.16	3.16
1	Electro Mart	16	Sony Corporation	2,015	8	August	90,956.01	84,755.97	7.32	7.32
1	Electro Mart	16	Sony Corporation	2,015	9	September	85,035.03	90,956.01	-6.51	6.51
1	Electro Mart	16	Sony Corporation	2,015	10	October	86,100.22	85,035.03	1.25	1.25
... 566 more rows omitted										

5.17 Q16: Top 5 Products Purchased Together Across Multiple Orders (Product Affinity Analysis)

Performance: 1.866s — **Rows Returned:** 0

This query returned no results. Analysis: The transactional dataset contains no orders with multiple products (each Order_ID has exactly one Product_Key), preventing product affinity analysis. This is a characteristic of the academic dataset, not a query logic error.

5.18 Q17: Yearly Revenue Trends by Store, Supplier, and Product with ROLLUP

Performance: 851.19ms — **Rows Returned:** 6,630

Table 16: Results for Q17

StoreName	SupplierName	Product_ID	Year	Total_Revenue	Total_Quantity
ALL STORES	ALL SUPPLIERS	ALL PRODUCTS		14,782,845.77	364,513
ALL STORES	ALL SUPPLIERS	ALL PRODUCTS	2,017	7,386,466.11	182,541
Electro Mart	ALL SUPPLIERS	ALL PRODUCTS	2,017	1,036,807.17	25,388
Electro Mart	Sony Corporation	ALL PRODUCTS	2,017	1,036,807.17	25,388
Electro Mart	Sony Corporation	P00000442	2,017	690.20	34
Electro Mart	Sony Corporation	P00001642	2,017	2,203.42	131
Electro Mart	Sony Corporation	P00001742	2,017	9,783.84	136
Electro Mart	Sony Corporation	P00002642	2,017	1,003.60	13
Electro Mart	Sony Corporation	P00003042	2,017	95.44	4
Electro Mart	Sony Corporation	P00003242	2,017	7,251.86	277
... 6,620 more rows omitted					

5.19 Q18: Revenue and Volume-Based Sales Analysis for Each Product for H1 and H2

Performance: 331.13ms — **Rows Returned:** 3,321

Table 17: Results for Q18

Product_ID	Product_Category	Year	H1_Revenue	H2_Revenue	Yearly_Revenue	H1_Quantity	H2_Quantity	Yearly_Quantity	H1_Revenue_Percentage	H2_Revenue_Percentage
P00059442	Household Essentials	2,017	15,166.86	19,260.57	34,427	226	287	513	44.05	55.95
P00110842	Grocery	2,017	18,108.72	15,018.74	33,127	252	209	461	54.66	45.34
P00220442	Health & Beauty	2,017	14,640.36	17,669.40	32,309	203	245	448	45.31	54.69
P00184942	Grocery	2,017	17,651.04	11,767.36	29,418	264	176	440	60	40
P00044442	Grocery	2,017	14,058.10	13,910.12	27,968	190	188	378	50.26	49.74
P00277642	Electronics	2,017	12,912.44	14,536.16	27,448	167	188	355	47.04	52.96
P00265242	Health & Beauty	2,017	13,891.28	11,620.96	25,512	361	302	663	54.45	45.55
P00025442	Grocery	2,017	12,444.66	11,826.84	24,271	282	268	550	51.27	48.73
P00034742	Health & Beauty	2,017	10,790.80	13,081.30	23,872	212	257	469	45.20	54.80
P00031042	Toys	2,017	12,103.52	11,521.62	23,625	208	198	406	51.23	48.77
... 3,311 more rows omitted										

5.20 Q19: Identify High Revenue Spikes in Product Sales and Highlight Outliers

Performance: 8.273s — **Rows Returned:** 15,492

Table 18: Results for Q19

Product_ID	Product_Category	Full_Date	Day_Name	Daily_Sales	Avg_Daily_Sales	Sales_Multiple	Anomaly_Flag
P00000142	Home & Kitchen	2016-11-03	Thursday	278	71.66	3.88	HIGH SPIKE
P00000142	Home & Kitchen	2019-05-29	Wednesday	222.40	71.66	3.10	HIGH SPIKE
P00000142	Home & Kitchen	2020-03-27	Friday	222.40	71.66	3.10	HIGH SPIKE
P00000142	Home & Kitchen	2019-08-17	Saturday	222.40	71.66	3.10	HIGH SPIKE
P00000142	Home & Kitchen	2016-07-19	Tuesday	222.40	71.66	3.10	HIGH SPIKE
P00000142	Home & Kitchen	2020-07-11	Saturday	222.40	71.66	3.10	HIGH SPIKE
P00000142	Home & Kitchen	2020-12-27	Sunday	222.40	71.66	3.10	HIGH SPIKE
P00000142	Home & Kitchen	2020-08-08	Saturday	194.60	71.66	2.72	HIGH SPIKE
P00000142	Home & Kitchen	2017-12-23	Saturday	194.60	71.66	2.72	HIGH SPIKE
P00000142	Home & Kitchen	2016-10-26	Wednesday	194.60	71.66	2.72	HIGH SPIKE
... 15,482 more rows omitted							

5.21 Q20: Create a View STORE$_{Q}UARTERLY_{S}ALESforOptimizedSalesAnalysis$

Performance: 2.444s — **Rows Returned:** 192

Table 19: Results for Q20

Store_ID	Store_Name	Year	Quarter	Quarter_Label	Quarterly_Sales	Quarterly_Quantity	Total_Transactions	Unique_Customers	Avg_Transaction_Value
1	Electro Mart	2,015	1	Q1-2015	253,146.39	6,374	3,178	1,981	79.66
1	Electro Mart	2,015	2	Q2-2015	252,884.22	6,363	3,187	1,964	79.35
1	Electro Mart	2,015	3	Q3-2015	260,747.01	6,376	3,169	1,993	82.28
1	Electro Mart	2,015	4	Q4-2015	265,966.55	6,438	3,229	2,018	82.37
1	Electro Mart	2,016	1	Q1-2016	257,007.60	6,387	3,224	1,997	79.72
1	Electro Mart	2,016	2	Q2-2016	256,755.96	6,355	3,180	1,969	80.74
1	Electro Mart	2,016	3	Q3-2016	264,720.63	6,403	3,235	2,009	81.83
1	Electro Mart	2,016	4	Q4-2016	258,873.43	6,372	3,177	1,946	81.48
1	Electro Mart	2,017	1	Q1-2017	266,109.81	6,592	3,248	1,992	81.93
1	Electro Mart	2,017	2	Q2-2017	252,927.52	6,221	3,125	1,949	80.94
... 182 more rows omitted									

5.22 Performance Summary and Analysis

The query execution results demonstrate that the star schema design and indexing strategy successfully support near-real-time analytical workloads:

Key Performance Insights:

- **95% of queries execute in under 3 seconds:** Suitable for interactive dashboards
- **Window function queries (Q9, Q12, Q15):** 0.24s - 1.70s despite complex calculations
- **Large result set queries (Q14):** 2.95s for 66,226 rows demonstrates efficient full-table scans
- **Outlier detection (Q19):** 8.27s due to per-product statistical aggregation over daily data
- **View creation (Q20):** 2.44s includes DDL execution + initial materialization

Indexing Strategy Impact: The composite indexes on (Date_Key, Product_Key), (Store_Key, Supplier_Key), and (Order_ID, Product_Key) enable efficient query plan optimization. Foreign key indexes on all dimension keys ensure sub-second joins for most queries.

Data Warehouse Scalability: With 550,068 fact records and total execution time under 30 seconds for all 20 queries, the system demonstrates production-ready performance for Walmart's near-real-time business intelligence requirements.

6 Conclusion and Lessons Learned

This project demonstrated end-to-end data warehousing: schema design, ETL implementation, and analytical query development. The HYBRIDJOIN algorithm [1] provided hands-on experience with stream processing challenges—bounded memory, disk I/O optimization, and multi-relation joins.

The evolution through five iterations showed that algorithm implementation is iterative. Understanding the theory from Naeem et al. [1] was necessary but not sufficient—practical concerns like batch I/O, memory bounds, and production scalability required experimentation.

6.1 Technical Skills Acquired

1. **Stream Processing Architecture:** Learned to design multi-threaded producer-consumer systems with proper synchronization using queues, events, and locks.
2. **Database Performance:** Understood importance of indexes, batch operations, and reducing round-trip queries. A single `IN` clause batch query can be 100× faster than loop-based individual queries.
3. **Algorithm Trade-offs:** Realized that "academic correctness" (chained HYBRIDJOIN) vs "practical optimization" (batch lookups) depends on problem constraints. Both have value—one for scalability guarantees, the other for performance in bounded scenarios.
4. **Memory Management:** Experienced how bounded data structures (fixed hash table, limited queue) enable predictable resource usage critical for production systems.

6.2 Problem-Solving Process

1. **Iteration is Key:** Went through five implementations before finding the right balance. Each failure taught something about constraints not initially considered.
2. **Measure, Don't Assume:** Initially thought row-by-row lookups would be "fast enough." Measuring revealed 100× inefficiency.
3. **Understand Requirements:** Took time to grasp why the instructor emphasized disk-based partitions. The point wasn't this specific dataset, but designing for scale.

6.3 Lessons Learned

Technical Skills:

1. **Stream Processing:** Designed multi-threaded producer-consumer systems with proper synchronization using queues and locks for bounded-memory guarantees.
2. **Database Optimization:** Batch operations (SQL `IN` clauses) are 100× faster than row-by-row queries. Composite indexes critical for OLAP performance.
3. **Memory Management:** Fixed-size hash tables and queues enable predictable resource usage for production systems.

Algorithm Trade-offs: Pure HYBRIDJOIN (Iteration 6: 36.78s) vs practical batch optimization (Iteration 4: 30s) depend on constraints. HYBRIDJOIN guarantees scalability for arbitrarily large master data; batch loading is faster for bounded datasets.

Data Warehousing Concepts:

1. **Star Schema:** Denormalized dimensions [3] simplify OLAP queries vs normalized OLTP schemas.
2. **ETL Complexity:** Data transformation (stream-relation joins) is harder than storage. Most warehouse complexity lives in the transformation layer.

3. **OLAP Operations:** Window functions (`LAG`, `RANK`, `NTILE`) enable powerful time-series analysis and segmentation.

Key Insight: The evolution through 6 iterations showed that effective architectures separate concerns: (1) bounded stream processing (HYBRIDJOIN maintains constant 32K tuples), (2) thin dimension lookups (76 KB for key mappings), (3) wide master enrichment (partitioned disk access). Conflating these concerns causes either memory overflow (Iterations 1-3) or performance degradation (Iteration 2: 310s).

7 References

1. Naeem, M.A., Dobbie, G., & Weber, G. (2011). HYBRIDJOIN for Near-Real-Time Data Warehousing. In: M. G. (eds) *Twenty-Second Australasian Database Conference (ADC 2011)*, Perth, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 115.

2. Codd, E.F., Codd, S.B., & Salley, C.T. (1993). Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate. *Codd & Date, Inc. Technical Report*.

3. Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (3rd ed.). John Wiley & Sons.