Table of Contents

Final Report

Team 16: Junyi Chen, Ananya Rattan Khurana, Fuqian Zou

1. Problem Statement

In today's competitive retail environment, traditional department stores like Dillard's face significant challenges due to the rise of e-commerce, shifting consumer preferences, and market volatility. This project leverages Dillard's extensive dataset—comprising transaction details, store information, SKU data, and department classifications—to predict sales amounts and enhance operational efficiency. The objectives are to:

- Identify key factors driving sales performance and their impact on revenue.
- Analyze demand fluctuations and trends across time, stores, and SKUs.
- Evaluate the effects of departments, brands, pricing, and timing on sales variability.
- Explore the role of SKU attributes and historical demand in shaping sales patterns.

2. Methodology

2.1 Data Loading

The dataset was loaded into a PostgreSQL database after connecting to the server, identifying five CSV files, and creating five database tables. Data was ingested hierarchically, with parent tables processed first and child tables loaded subsequently to preserve foreign key dependencies. Inconsistencies, such as special character parsing and mismatched column orders, were resolved using SQL functions, column reordering, and the addition of boolean flags to address extra columns (code for loading data could be found in Appendix A).

2.2 Data Quality Checks

Quality checks ensure data accuracy and consistency. Record counts were validated between server files and database tables, while joins between related tables were verified to maintain relational integrity. Basic statistical analyses and SQL checks were conducted to validate column reliability, identify anomalies, and ensure data consistency. This included detecting missing values, verifying numerical ranges (e.g., negative or zero values), checking data relationships

(e.g., transaction amounts aligning with quantities and prices), and validating categorical values against expected enumerations (code for checking data could be found in Appendix B).

2.3 Challenges and Solutions

Challenges included handling column mismatches, inconsistent data structures, and duplicate entries. These issues were resolved by reordering columns in CSV files, resolving duplicate `amt` columns, parsing complex records with an SQL function for the `skuinfo` table, and adding an additional `bool` column where necessary to align with the database schema.

2.4 Tools and Technologies

PostgreSQL and pgAdmin were utilized for database operations. Python libraries such as pandas and sqlalchemy facilitated data manipulation and integration with the database. Custom SQL queries optimized data extraction and table joins to ensure efficient workflows.

3. Exploratory Data Analysis

Our exploratory data analysis aimed to uncover patterns and trends to guide modeling and strategy development.

Aspect	Output	Insights
Top 20 Brands (SKUs)	Analysis of SKU numbers shows a significant concentration in the top brands, like POLO FAS.	A few brands dominate SKU distribution, requiring tailored inventory strategies.
Purchase vs. Returns	Purchases make up 97.3%, while returns are only 2.7%, indicating a low return rate.	A low return rate indicates strong customer satisfaction and efficient return policies.
Descriptive Statistics	Quantity: Min: 1, Max: 90, Avg: 1.000, Price: Min: 0, Max: 6017, Avg: 24.62 Amount: Min: 0, Max: 6017, Avg: 24.62	High price variability suggests a diverse product portfolio, while stable averages show consistent sales.
Monthly Transactions	The highest monthly transaction count reached **1,200,000**, with total monthly amounts peaking at **$60M**.	Peaks during holidays highlight opportunities for targeted seasonal promotions and inventory adjustments.
Top 10 Departments	High performers include CLINIQUE ($244.7M), POLOMEN ($227.6M), and CELEBRT ($165.5M).	Investing in these top departments could further enhance revenue and profitability.

Top 10 States	The top state(TX) recorded **$300M in sales**, nearly double the **$160M** of the second-ranked state(FL).	Focused regional marketing in top-performing states could maximize sales potential.

4. Feature Engineering

The goal of this project is to predict monthly sales based on SKU attributes and store information. The response in the model is the sum of sales grouped by month, SKU, and store. Predictors are month, SKU attributes, and store information. We selected or created the following variables as predictors:

Feature Name	Feature Type	Features Description
Cost	Numeric	The cost of the stock item.
Retail	Numeric	The retail price of the stock item.
Packsize	Numeric	The quantity of items per pack.
Month	Categorical	Converted saledate to DateTime type, extracted month from saledate, and dropped data in August 2025 to avoid double counting monthly sales in August.
Color_category	Categorical	The color of the stock item. Most frequent colors were categorized to major types (black, white, red, blue, etc.).
State	Categorical	State where the store is located.

For numeric features, values were standardized. Numeric variables were scaled to have a mean of 0 and a standard deviation of 1. It ensured that no feature dominates others due to its scale. For categorical features, values were one-hot encoded. Categorical variables were converted into a numerical format by creating binary (0/1) columns for each category, which enabled models to interpret and utilize categorical information effectively.

5. Modeling

5.1 Model Selection

In this project, we utilized a variety of models to predict sales. Multiple linear regression was chosen for its simplicity and interpretability, which can provide insights into linear relationships between features and sales. Then we chose ridge regression that incorporates L2 regularization to mitigate overfitting and manage multicollinearity among predictors. Similarly, lasso regression

with L1 regularization was used for feature selection and model simplification. To capture non-linear relationships and improve computational efficiency on large datasets, we used two gradient boosting models, LightBGM and XGBoost.

5.2 Model Training and Evaluation

Linear Models

- Linear Regression: $R^2 = 0.6312$
- Ridge (α=100): $R^2 = 0.6312$
- Lasso (α=0.01): $R^2 = 0.6310$

Gradient Boosting

- LightGBM: $R^2 = 0.7575$ (best performer)
- XGBoost: $R^2 = 0.7532$

Model	Mean Squared Error (MSE)	R-squared (R^2)
Linear Regression	292.91	0.6312
Ridge Regression	292.91	0.6312
Lasso Regression	293.05	0.6310
LightGBM	192.57	0.7575
XGBoost	195.99	0.7532

Key finding: The model evaluation results show ensemble methods outperforming traditional linear approaches by approximately 12 percentage points in R^2, suggesting that future implementations should prioritize non-linear modeling techniques for similar retail transaction prediction systems.

6. ROI analysis and Financial Potential

The financial analysis showcases the impact of the sales prediction model, generating an annual net gain of **$153.5 million**. Powered by the LightGBM algorithm (R2=0.7575), the model reduces overstock and stockout rates by 50%, translating into substantial cost savings and improved revenue accuracy.

The project requires an annual investment of **$560,347**, including **$2,847** for technology (cloud computing and storage) and **$557,500** for labor. With this modest investment, the model delivers an exceptional **ROI of 27,400.20%**, breaking even within the first year. (Details in Appendix-ROI Analysis)

7. Recommendations for Generating Business Value

7.1 Reduced Inventory Costs

Accurate demand forecasting reduces overstock, cuts storage costs, minimizes waste, and protects profit margins by optimizing inventory levels.

Recommendation: Dynamically allocate inventory based on demand predictions to avoid excess stock and ensure availability of high-demand items.

7.2 Increased Revenue

Ensuring the right SKUs prevents lost sales, supports cross-selling and upselling, and allows strategic promotions for high-demand SKUs to maximize revenue during peak periods.

Recommendation: Use targeted promotions—discount low-demand SKUs to drive sales while optimizing pricing strategies for high-demand SKUs to maximize revenue.

7.3 Improved Efficiency

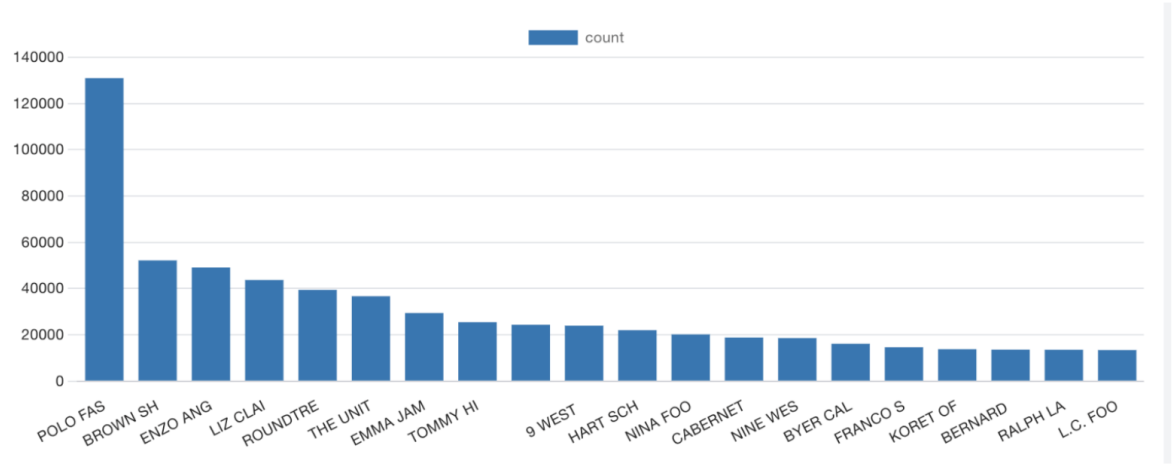
Automated sales prediction reduces errors in inventory planning and procurement, enabling teams to focus on strategic tasks. Enhanced demand visibility streamlines supply chain operations, improving vendor coordination and logistics.

Recommendation: Integrate predictive models with real-time data to ensure continuous optimization of inventory and promotional strategies, aligning with dynamic demand trends.

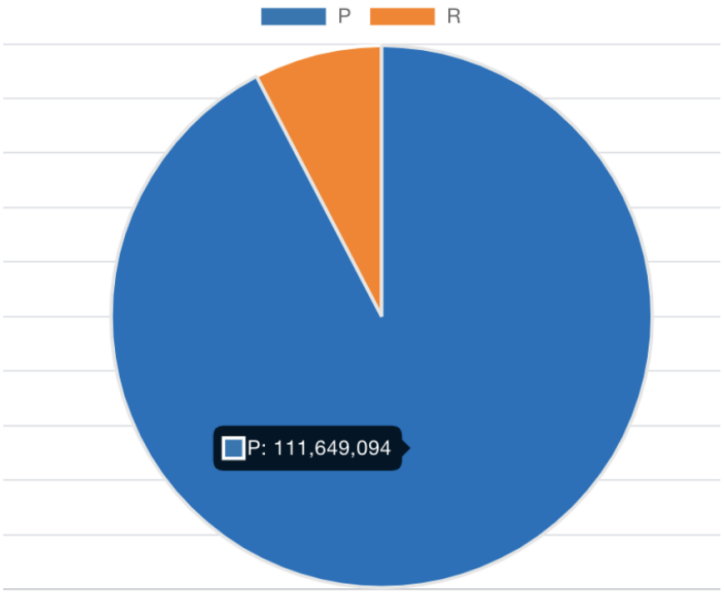
Appendix

EXPLORATORY DATA ANALYSIS

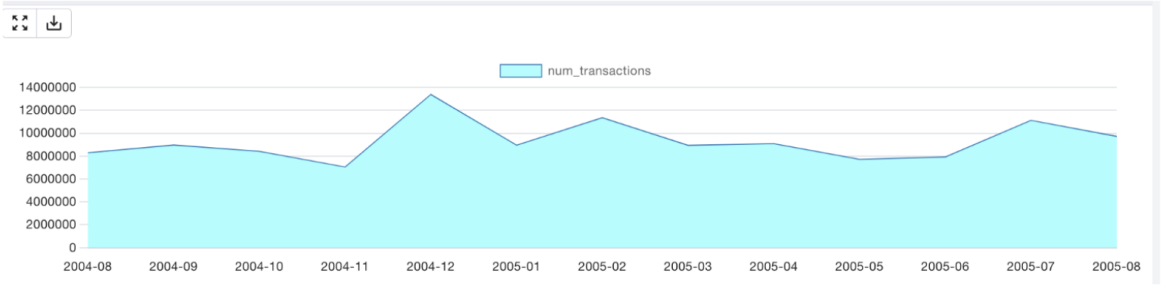
TOP 20 BRANDS BY SKU

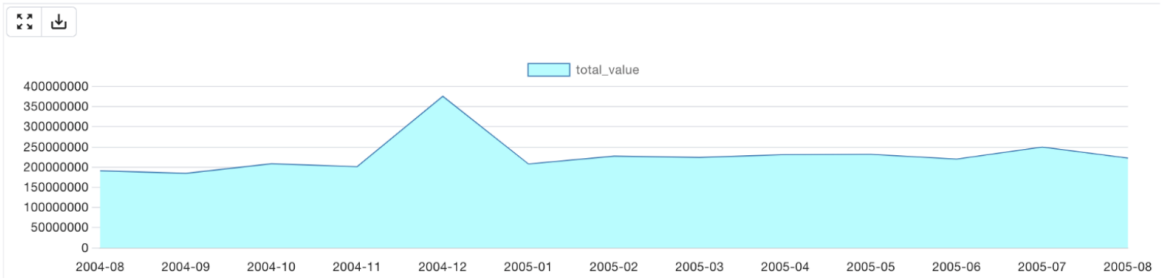


PURCHASE AND RETURN DISTRIBUTION

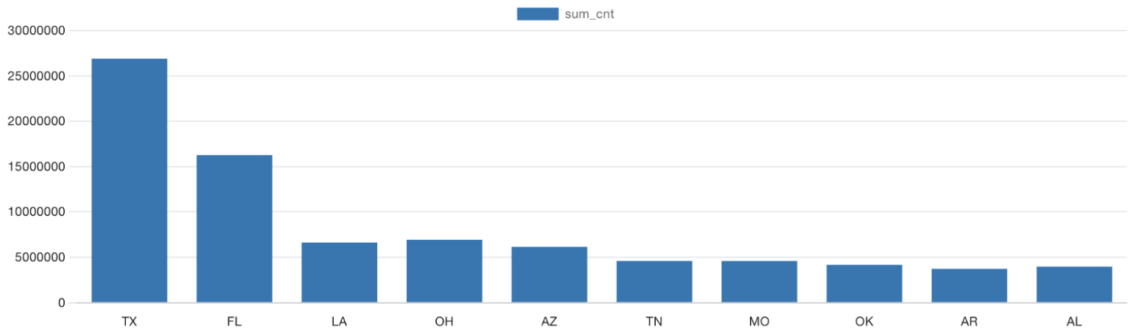


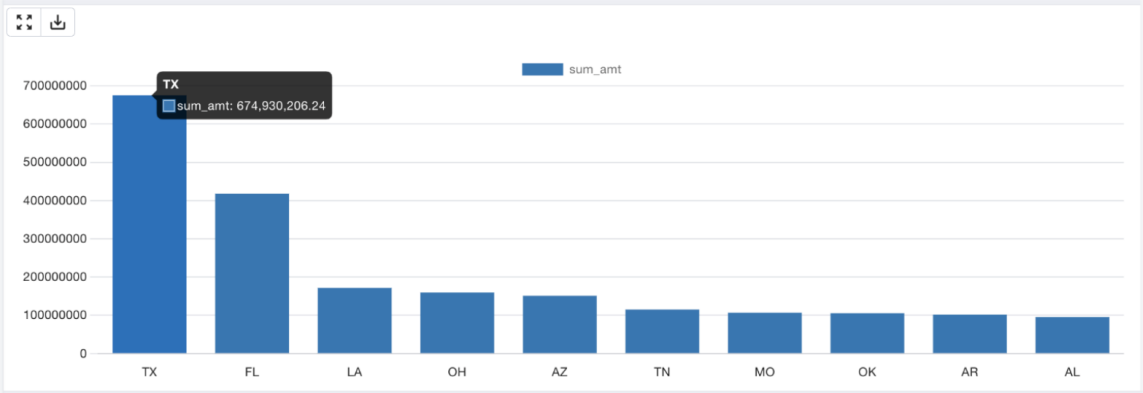
MONTHLY DISTRIBUTION OF TRANSACTION COUNT AND TOTAL AMOUNT





TOP 10 STATES BY TRANSACTION COUNT AND AMOUNT





MAIN CODES

A. Data Loading code

```
## Load data

# we create tables based on the schema and in the interactive page in pgadmin4.

• deptinfo

# server
\copy team16.deptinfo (dept, deptdesc, bool) from '/nfs/home/dillards/deptinfo.csv' WITH
DELIMITER ',' CSV;

• strinfo

# server
\copy team16.strinfo (store, city, state, zip, bool) from '/nfs/home/dillards/strinfo.csv' WITH
DELIMITER ',' CSV QUOTE ‘“';

• skuinfo

# server
# serialize datasets, add '\' before '\' in data to ensure it will not be lost when loading.
sed 's/\\/\\\\/g' skuinfo.csv >skuinfo_test.csv

# load into a tmp table in one column for further cleaning in SQL since it has commas in the
dataset
\COPY team16.tmp FROM ‘/nfs/home/hzc8492/skuinfo_test.csv’ WITH (FORMAT text);

# SQL
-- Use string_to_array, position, split_part function in SQL to extract the desired column and
load into skuinfo
set standard_conforming_strings=on;

insert into team16.skuinfo
select
cast((string_to_array(a, ','))[1] as integer) as i
,cast((string_to_array(a, ','))[2] as integer) as i2
,cast((string_to_array(a, ','))[3] as varchar) as i3
,cast((string_to_array(a, ','))[4] as varchar) as i4
,cast(substring(a ,position((string_to_array(a, ','))[4] in a)
+ char_length((string_to_array(a, ','))[4])+1 ,12) as varchar) as i5
```

```
,cast(substring(a ,position((string_to_array(a, ','))[4] in a)
+ char_length((string_to_array(a, ','))[4])+14 ,12) as varchar) as i6
,cast(substring(a ,position((string_to_array(a, ','))[4] in a)
+ char_length((string_to_array(a, ','))[4])+27 ,10) as varchar) as i7
,cast(split_part(substring(a,position(substring(a ,position(SPLIT_PART(a,',',4) in a)
+ char_length(SPLIT_PART(a,',',4))+1,10) in a) + 37),',',1) as integer) as i8
,cast(split_part(substring(a,position(substring(a ,position(SPLIT_PART(a,',',4) in a)
+ char_length(SPLIT_PART(a,',',4))+1,10) in a) + 37),',',2) as varchar) as i9
,cast(substring(a,position(split_part(substring(a,position(substring(a
,position(SPLIT_PART(a,',',4) in a)
+ char_length(SPLIT_PART(a,',',4))+1,10) in a) + 37),',',2) in a)
+ length(split_part(substring(a,position(substring(a ,position(SPLIT_PART(a,',',4) in a)
+ char_length(SPLIT_PART(a,',',4))+1,10) in a) + 37),',',2))+1,9) as varchar) as i10
,cast((string_to_array(a, ','))[array_length(string_to_array(a, ','), 1)] as integer) as i11
from team16.tmp;
```

• skstinfo

```
# server
\copy team16.skstinfo (sku, store, cost, retail, bool) from ‘/nfs/home/dillards/skstinfo.csv' WITH
DELIMITER ',' CSV;
```

• trnsact

```
# server
\copy team16.trnsact (sku, store, register, trannum, interid, saledate, stype, quantity, orgprice,
amt, amt2,seq, mic, bool) from '/nfs/home/dillards/trnsact.csv' WITH DELIMITER ',' CSV;
```

B. Data check code

1. records count check

```
# server
find . -name '*.csv' | xargs wc -l
  120916896 ./trnsact.csv *
         60 ./deptinfo.csv *
   39230146 ./skstinfo.csv *
        453 ./strinfo.csv *
    1564178 ./skuinfo.csv *
  161711733 total
```

```
# SQL
select count(*) from team16.deptinfo;
-- 60

select count(*) from team16.strinfo;
-- 453

select count(*) from team16.skuinfo;
--  1564178

select count(*) from team16.skstinfo;
-- 39230146

select count(*) from team16.trnsact;
--  120916896
```

2. Data quality check

```
-- 1. Check for missing or null values:

SELECT
   'SELECT ''' || table_name || ''' AS table_name, ' ||
   'COUNT(*) AS total_rows, ' ||
   string_agg('SUM(CASE WHEN ' || column_name || ' IS NULL THEN 1 ELSE 0 END) AS
missing_' || column_name, ', ') ||
   ' FROM ' || 'team16.' || table_name || ';'
FROM information_schema.columns
WHERE table_schema = 'team16'
GROUP BY table_name;

SELECT 'deptinfo' AS table_name, COUNT(*) AS total_rows, SUM(CASE WHEN deptdesc IS
NULL THEN 1 ELSE 0 END) AS missing_deptdesc, SUM(CASE WHEN bool IS NULL
THEN 1 ELSE 0 END) AS missing_bool, SUM(CASE WHEN dept IS NULL THEN 1 ELSE 0
END) AS missing_dept FROM team16.deptinfo;

SELECT 'skstinfo' AS table_name, COUNT(*) AS total_rows, SUM(CASE WHEN cost IS
NULL THEN 1 ELSE 0 END) AS missing_cost, SUM(CASE WHEN bool IS NULL THEN 1
ELSE 0 END) AS missing_bool, SUM(CASE WHEN retail IS NULL THEN 1 ELSE 0 END)
AS missing_retail, SUM(CASE WHEN store IS NULL THEN 1 ELSE 0 END) AS
missing_store, SUM(CASE WHEN sku IS NULL THEN 1 ELSE 0 END) AS missing_sku
FROM team16.skstinfo;
```

```
SELECT 'skuinfo' AS table_name, COUNT(*) AS total_rows, SUM(CASE WHEN style IS NULL THEN 1 ELSE 0 END) AS missing_style, SUM(CASE WHEN upc IS NULL THEN 1 ELSE 0 END) AS missing_upc, SUM(CASE WHEN classid IS NULL THEN 1 ELSE 0 END) AS missing_classid, SUM(CASE WHEN bool IS NULL THEN 1 ELSE 0 END) AS missing_bool, SUM(CASE WHEN sku IS NULL THEN 1 ELSE 0 END) AS missing_sku, SUM(CASE WHEN packsize IS NULL THEN 1 ELSE 0 END) AS missing_packsize, SUM(CASE WHEN dept IS NULL THEN 1 ELSE 0 END) AS missing_dept, SUM(CASE WHEN brand IS NULL THEN 1 ELSE 0 END) AS missing_brand, SUM(CASE WHEN vendor IS NULL THEN 1 ELSE 0 END) AS missing_vendor, SUM(CASE WHEN size IS NULL THEN 1 ELSE 0 END) AS missing_size, SUM(CASE WHEN color IS NULL THEN 1 ELSE 0 END) AS missing_color FROM team16.skuinfo;

SELECT 'strinfo' AS table_name, COUNT(*) AS total_rows, SUM(CASE WHEN city IS NULL THEN 1 ELSE 0 END) AS missing_city, SUM(CASE WHEN store IS NULL THEN 1 ELSE 0 END) AS missing_store, SUM(CASE WHEN zip IS NULL THEN 1 ELSE 0 END) AS missing_zip, SUM(CASE WHEN bool IS NULL THEN 1 ELSE 0 END) AS missing_bool, SUM(CASE WHEN state IS NULL THEN 1 ELSE 0 END) AS missing_state FROM team16.strinfo;

SELECT 'trnsact' AS table_name, COUNT(*) AS total_rows, SUM(CASE WHEN trannum IS NULL THEN 1 ELSE 0 END) AS missing_trannum, SUM(CASE WHEN seq IS NULL THEN 1 ELSE 0 END) AS missing_seq, SUM(CASE WHEN interid IS NULL THEN 1 ELSE 0 END) AS missing_interid, SUM(CASE WHEN quantity IS NULL THEN 1 ELSE 0 END) AS missing_quantity, SUM(CASE WHEN orgprice IS NULL THEN 1 ELSE 0 END) AS missing_orgprice, SUM(CASE WHEN amt IS NULL THEN 1 ELSE 0 END) AS missing_amt, SUM(CASE WHEN amt2 IS NULL THEN 1 ELSE 0 END) AS missing_amt2, SUM(CASE WHEN bool IS NULL THEN 1 ELSE 0 END) AS missing_bool, SUM(CASE WHEN register IS NULL THEN 1 ELSE 0 END) AS missing_register, SUM(CASE WHEN mic IS NULL THEN 1 ELSE 0 END) AS missing_mic, SUM(CASE WHEN saledate IS NULL THEN 1 ELSE 0 END) AS missing_saledate, SUM(CASE WHEN stype IS NULL THEN 1 ELSE 0 END) AS missing_stype, SUM(CASE WHEN sku IS NULL THEN 1 ELSE 0 END) AS missing_sku, SUM(CASE WHEN store IS NULL THEN 1 ELSE 0 END) AS missing_store FROM team16.trnsact;


-- 2. Check transact

-- Check for negative transaction amounts:
-- 0
```

```
SELECT count(*)
FROM team16.trnsact where amt2<0;

-- Check for negative or zero quantity:
--  0
SELECT count(*)
FROM team16.trnsact where quantity<1;

-- Check for transaction amounts=quantity*price:
--  0
SELECT count(*)
FROM team16.trnsact where amt2-quantity*amt>0.01;

-- Check for sale dates range:
-- "2004-08-01"          "2005-08-27"
SELECT min(SALEDATE), max(SALEDATE)
FROM team16.trnsact;

-- Check for invalid values in enumerated columns:
--  0
SELECT count(*)
FROM team16.trnsact
WHERE stype NOT IN ('P', 'R');

-- 3. Check skstinfo

-- Check for negative cost:
-- 0
SELECT count(*)
FROM team16.skstinfo where cost<0;

-- Check for negative retail:
--  0
SELECT count(*)
FROM team16.skstinfo where retail<0;

-- 4. Check skuinfo
-- Check for brand name:
select brand,count(*)
FROM team16.skuinfo
```

```
group by 1;

-- There exist some goods that do not have a brand, but we will not use them as the variable
-- “brand"       "count"
-- “          "       24356
```

ROI Analysis

Net Gain		Base Case	Model Case
Gain Type	**Detailed Content/Assumption**	**Annual Amount**	**Annual Amount**
(+)Estimate Revenue Gains	Average transaction amount: **$24.62** # of transaction: **120916896** Purchase rate: **97.3%**	2,896,595,682	2,896,595,682
(-)Sales Cost	Average cost for stock item: **$15.92**	-1,873,022,066	-1,873,022,066
(+)Quantify Cost Savings	Overstock: **20%** saved rate: **50%**(=1-(1-R^2)^0.5)	0	102,357,362
(+)Quantify Cost Savings	Stockout: **10%** saved rate: **50%**	0	51,178,681
Total Annual Benefits	Sum of annual gain.	**1,023,573,616**	**1,177,109,659**

Investment		
Technology-SaaS		**Annual Amount**
Cloud Computing cost	Cloud Computing Hour = 8 hours/day × 260 business days =**2080** hours Assume using AWS EC2 t3.medium instances at **$0.0416/hour** for the US East region	86.5280
Storage cost	Assume **10 TB** of storage for storing historical and operational sales data Assume using AWS S3 (Standard Storage).	2,760

	The cost is **$23/TB per month.**	
Total Technology Investment	Sum of computing and storage costs for sales prediction activities.	**2,847**

Labor		Annual Amount
FTE (full time employees)	Assume **10** retail **employees**, each earning an average of **$40,000/year,** supporting operations.	400,000
Technology Engineer	Assume 1 engineer responsible for maintaining cloud systems and sales prediction tools.	75,000
Data Scientist Teams	Assume **3 data scientists** working for **three months**, each earning **$110,000/year.**	82,500
Total Labor Investment	Sum of all labor costs required for retail operations and sales prediction support.	**557,500**

Calculations (Year)		Net Profit
Total Net Profit	Net gains after applying model	153,536,042
Total Investment	Sum of technology costs and labor costs	560,347
ROI (%)	Return on investment calculated as (Total Net Profit / Total Investment) * 100	**27400.20%**

Insights	
- High ROI highlights the profitability of this project.	
- LightGBM model	

accuracy (R-squared: 0.7575) supports strong benefits.	