

## Real Time Sales Forecasting

Guided by: Ajay Pratap

<sup>1</sup>YASH TYAGI, <sup>2</sup>SHIVANSH TIWARI

<sup>1,2</sup>Department of Computer Science and Engineering (Artificial Intelligence) IIMT College Of Engineering, Greater Noida

Email: <sup>1</sup> yashtyagi1402@gmail.com, <sup>2</sup> shivansht656@gmail.com

**Abstract:** The project will offer a full sales and inventory management system in real time. It is backed by PostgreSQL and automated using a strong Python-based backend. The system is built to assist retailers in managing future sales trends in inventory and forecasts, offering a full data pipeline from prognosis and visualization work and preprocessing. PostgreSQL ensures scalable and reliable data storage and facilitates high-performance queries using optimization methods like indexing and query limiting. These optimizations are designed to offer real-time awareness of dashboards and reports and a smooth user experience. These scripts perform data cleaning, such as double removal, formation standardization, zero value handling, and field validation, to ensure high data integrity. Upon sanitizing, produce a 12-month sales transaction based on information from forecasting models like exponential smoothing models. Forecasts enable organizations to anticipate trend patterns in demand, reduce attacks and overstocking, and streamline their selling plans. One can upload CSV files visualizing forecast costs and download processed data records through this interface. Dynamic diagrams and charts enable users to track inventory and sales trends in real time. These dashboards are built for decision-makers and offer visualizations of performance metrics, historical trends and predictive analytics at one glance. It is a compelling tool for businesses operating in the data-rich retail culture, allowing for proactive inventory management, data control decisions, and strategic demand planning.

**Keywords:** Real-Time Inventory Management, Sales Forecasting, PostgreSQL, Python Automation, Data Cleaning, Time Series Forecasting, Flask Web Application, Power BI Integration, Inventory Optimization, Business Analytics

### I. INTRODUCTION

In today's fast and competitive markets, effective sales monitoring and inventory tracking has become a business activity for businesses to ensure operational stability, profitability and customer satisfaction. With interest in real information and predictive analytics, archaic practices such as manual share management and retained sales no longer meet today's requirements. This project demonstrates the creation and delivery of a powerful, automated, real-time tracking and inventory management system. The system is developed with Python for automation and data processing, PostgreSQL as a backend relational database, flasks for web application interfaces, and Power BI for interactive business analysis and visualization.

The main goal of this project is to present an integrated platform where companies can view real-time sales data, pursue inventory, and predict future demand with less human intervention. It properly organizes tables for products, sales transactions, and forecast results. The product table contains information such as product ID, name, category, unit price, stock level, etc., while the sales table pursues all sales in data fields such as transaction ID, product ID, sales quantity, sales date, customer region, etc. The forecast table includes time series model issues such as index smoothing and sales patterns for up to 12 months. The platform has a solid data cleaning layer that maintains data consistency and accuracy by eliminating duplicate datasets, data type modifications, date format standardization, lack of value creation, and field entry validation. This clean data is stored in a PostgreSQL database and serves as a basis for forecasting and reporting. The core element of the system is Statsmodels - a prediction engine that uses the exponential smoothing model of the Python library. These

models consume historical data and predict future demand, including trends and seasonality.

The predicted output is stored in the database and displayed via the Flask Web usage and Power-Bi dashboard. Forecasting capabilities allow companies to predict stock purchases, price management and advertising campaigns. The user portion of the system is encoded in Flask and SQLAlchemy, which encourages dynamic engagement with the database and provides a user-friendly user interface. With the help of this web app, users can enter new CSV files, query current stock status, download mild data records, and visualize sales forecasts on combined diagrams. The interface also supports search and filtering features, allowing users to query the database for product names, categories, date ranges, or sales services.

This ensures detailed control over the data that users export or analyze. The software is quickly optimized and is used for desktop and mobile use, making it suitable for use with managers on the go. Power BI establishes a direct connection to a PostgreSQL database and picks up RAW and predictive data in real time. It offers a wide range of visualizations such as bar diagrams, linen diagrams, KPIs and more. The Power BI Dashboard facilitates investigation of previous trends in dynamic filtering, holes in specific data segments, and predictions. For example, you can view monthly sales according to product categories, check local sales performance, and reduce demand for current stocks with prognosis. These features allow for deeper insight into business activities and help you determine where you can maximize efficiency. This system is especially useful in retail environments where sales speeds are high and require quick scales to prevent excess inventory and inventory.

## II. LITRATURE REVIEW

The emergence of supply chain management and the modern world of retailing have made it essential to integrate into real-time inventory and sales dashboards for operational transparency and decision-making effectiveness. The ability to track sales trends, stocks and demand patterns in flight not only improves daily operations but also forms the basis for long-term strategic planning. These dashboards are the central nervous system of data-controlled companies, providing interest groups with a single view of key metrics, driving faster answers to changing market dynamics and improviser risks. PostgreSQL, an open-source relational database management system, is centrally important for managing large amounts of transactional data with high consistency and performance. Skills such as sophisticated indexing mechanisms, data limits, and real-time query support make them ideal for managing sales and inventory data. By enforcing data restrictions and maintaining consistency through referential integrity, PostgreSQL allows for accurate recording of data. This is important importance for downstream analysis. However, data alone is not sufficient. The quality of conclusions that can be created depends on the quality of the data processed. In this respect, Python plays a very powerful Midssman role in the data pipeline. It is commissioned to clean and design raw unstructured data in a structured format suitable for analysis. In modules like Pandas, NumPy, and Scikit-Learn, Python scripts perform important pre-processing processes such as data definition, missing values assignment, data type conversion, and outlier detection.

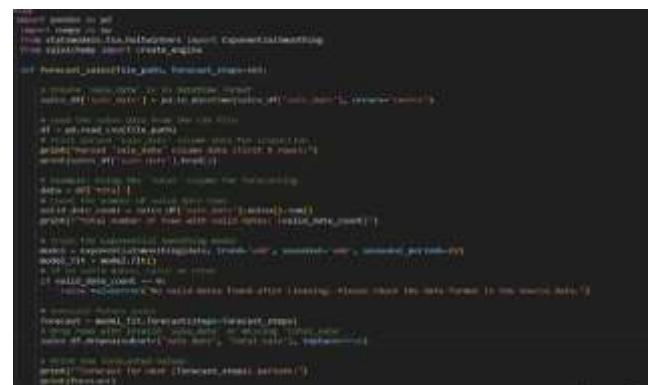
These processes allow the analytical model to receive clean, well-formatted data, improving prediction or pattern reliability. Another important aspect of this system is prediction. Retail environments typically have cyclical demand patterns due to seasonality, advertising campaigns and market trends. Prediction techniques such as exponential smoothing (ETS) are used to take into account these dynamics. Due to its simplicity and strength, the model is suitable for creating stable predictions in situations where the latest patterns and seasonality are very important. By integrating these models into the pipeline, the system can predict inventory requirements, allowing businesses to minimize inventory and reduce hearing costs. Light Python -Web -Framework Flask is used as a backend to handle API requirements, database connections, and routing. It provides uniform integration between the front-end dashboard and the analytics engine. However, Power BI is used to develop dynamic visualizations with drill-down capabilities, real-time filtering, and KPI monitoring. These dashboards are not only visual tools, but also decision-making systems that help users recognize anomalies, monitor performance indicators, and detect trends in consumer behavior. This pipeline continuously absorbs and processes new data from sales and existing systems, updates forecasts, and updates dashboards in real time. This automation allows decision makers to always have access to the latest data without manual intervention. It improves agility and responsiveness, allowing businesses to act proactively and reactively. Most systems focus solely on data analysis or solely on data visualization. By combining these two aspects into a single complete solution, this project excludes this gap. Dashboards don't just show you previous data. It provides future information and downloadable analysis that allows for further testing and reporting.

In the real world, there are many advantages of such a system. Prices can be dynamically changed depending on inventory and expected demand. The procurement plan is redesigned by the supply chain manager, which allows managers to assess the orientation of their strategic targets using real-time KPIs. The modular architecture of the solution allows it to be scaled or customized to accommodate a wide range of industries, from healthcare supply chains to e-commerce bakeries. In summary, this project illustrates an end-to-end approach and sales tracking. Full integration of open-source tools searches as Power BI and Flask using PostgreSQL and Python. It demonstrates how to harmonize and integrate automated pipelines, real-time dashboards and predictive analytics to improve operational effectiveness and strategic decision-making. Focusing on data quality, forecast accuracy and interactivity, the system places the prerequisites for more intelligent, faster, and better-informed decision-making in today's dynamic trade world.

## III. METHODOLOGY

### A. DATA CLEANING:

The initial step in the data processing pipeline involves the use of a specialized Python module called `data_cleaning.Py`, which is responsible for ensuring the accuracy and consistency of the dataset before it moves forward in the pipeline. One of the main purposes of this module is identifying and removing duplicate records based on invoice IDs and unit prices. By removing redundant entries, this prevents any potential distortions in analytical models or visual representations. Furthermore, the module standardizes data formats, handles missing values, and validates field inputs to guarantee that only clean and properly formatted data is transmitted. This process is essential for achieving accurate forecasting and reliable reporting. phase.



```
import pandas as pd
import psycopg2
from sqlalchemy import create_engine
from datetime import datetime

def connect_to_db(host, user, password, dbname):
    """Establish a connection to the PostgreSQL database"""
    engine = create_engine(f'postgresql://{user}:{password}@{host}/{dbname}')
    return engine

def load_data(engine):
    """Load data from the database into a DataFrame"""
    query = "SELECT * FROM sales_data"
    df = pd.read_sql(query, engine)
    return df

def clean_data(df):
    """Clean the data by removing duplicates and handling missing values"""
    # Remove duplicates based on invoice_id and unit_price
    df = df.drop_duplicates(subset=['invoice_id', 'unit_price'], keep='first')

    # Handle missing values
    df['unit_price'] = df['unit_price'].fillna(0)
    df['quantity'] = df['quantity'].fillna(0)

    # Convert data types
    df['invoice_id'] = df['invoice_id'].astype(str)
    df['unit_price'] = df['unit_price'].astype(float)
    df['quantity'] = df['quantity'].astype(int)

    return df

def save_data(df, engine):
    """Save the cleaned data back to the database"""
    df.to_sql('cleaned_sales_data', engine, if_exists='replace', index=False)
```

FIGURE 1

### B. RECORDING DATA :

Preprocessing The PostgreSQL database receives cleaned data through a module called (`data_ingress.Py`), which is specifically designed for data ingestion. The module ensures that accurate and non-redundant data is inputted into the system. During the registration process, the module first checks for any existing records by comparing compound keys such as `invoice_id` and `sale_date` to avoid duplications. This validation step becomes crucial for maintaining the consistency of transactional data and also guaranteeing the preservation of older data. New information and resources are added to the appropriate tables—products and sales—within the database. This continuous process of inserting data on the fly allows the system to

```
# Monte Carlo steps for forecasting inventory
# Will be a distribution model in calculating the average monthly
avg_inventory = 40 * (inventory / 10000)
# Forecast for the next 12 months
forecast_sales = model.fit(forecast_steps)

# Forecast for the next 12 periods as an example
forecast = get_forecast_random_forest_model(avg_inventory, 12)
print('Inventory forecast for next 12 periods:')
print(forecast)

# Output Forecast
# Create a distribution for forecasting inventory
forecast_sales_df = pd.DataFrame()
forecast_sales_df['date'] = range(forecast_start, forecast_end, 1)
forecast_sales_df['avg_inventory'] = forecast_sales.values

# Monte Carlo steps for forecasting demand
# This is a distribution model in calculating the average monthly
avg_demand = 40 * (inventory / 10000) * (forecast_sales / 10000)
# Calculate a rolling average
# Forecast for the next 12 periods as an example
forecast = get_forecast_random_forest_model(avg_demand, 12)

print('Demand forecast for next 12 periods:')
print(forecast)
return forecast_sales_df
```

### C. SALES FORECAST

The forecasting component of the system is implemented in the (data\_forecasting.Py) module and is crucial for the delivery of predictive analytics. The module utilizes exponential index smoothing, a forecasting technique that takes into account both the overall trend and seasonal patterns in sales data, to predict future demand for different products. It is equipped with historical sales data stored in the PostgreSQL database, enabling it to analyze past performance trends and make predictions about future results. The forecasting algorithm generates monthly sales predictions for each product over a 12-month timeframe. These predictions are stored in a dedicated table within the database, allowing for convenient retrieval and examination for additional analysis and visualization purposes. The projected data is beneficial for a range of business applications, including stock planning, optimizing pricing strategies, and matching campaigns effectively. Additionally, the module structure allows for continuous learning with the new data becoming available, leading to the regular updating of the forecasting model and ensuring its relevance and effectiveness in a dynamic market.

The web server module of the system is constructed using flask in the (app.Py) file to create a lightweight and interactive interface for delivering and displaying data. The flask application incorporates specific routes to present real-time sales and product tables directly from the postgresql database. The routes allow users to access real-time data using a web browser. Sqlalchemy is employed as the object-relational mapping (orm) utility to ensure seamless and efficient communication between the postgresql backend and the flask application. Pandas is also employed to convert raw query results into organized dataframes, simplifying the process of rendering and exporting data in CSV format.

Power BI is seamlessly connected to the postgresql database, allowing for real-time visual analysis of both historical and projected sales data. This direct link enables the dashboard to display real-time updates as new information is received or forecasts are created. The power bi dashboard is engineered with full interactivity, providing advanced features like the ability to delve into specific regions, product categories, or time periods. Users have the ability to apply dynamic filters, analyze trends, compare performance across different groups, and export filtered views for detailed offline reporting. These capabilities enable business stakeholders to conduct thorough investigations, uncover patterns, and make informed decisions based on data, ensuring confidence and efficiency.

[illegible]

### FIGURE 4

```

# Create Inventory Forecast df
# Save the forecast data to file as dictionary, first row is header, saving to csv
df = {'ForecastIndex': Forecast_idx, 'Inventory Forecast df':
      sales_forecast_df.to_csv('sales_forecast.csv', index=False),
      'Inventory Forecast df to csv': Inventory_Forecast_csv, 'ActualSales':
      print('Forecasts saved successfully.')}

# Split Actuals in Forecast index and Inventory
df['ForecastIndex'] = Forecast_idx
df['ForecastIndex'] = Forecast_idx

# Split sales data from database
sales_df = pd.read_sql('SELECT * FROM sales', engine)

# Split the data into two of the new database to ensure data is loaded correctly
print('New sales database created from the database')
projection_df['Sales']

# If sales df is empty, raise an error
if sales_df.empty:
    raise ValueError('Sales data is empty. Please check our data source or database connection.')

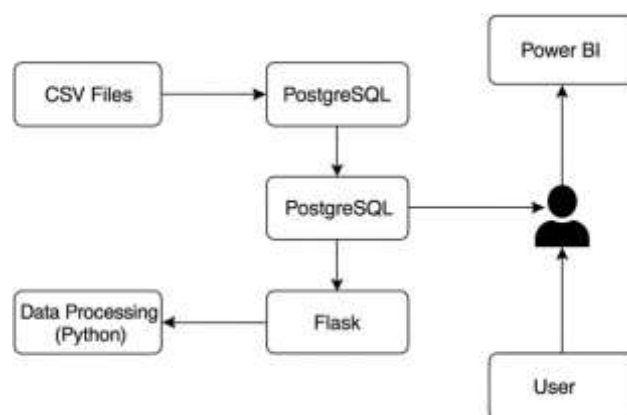
# Create Forecast
# Replace null values forecasting
sales_forecast_df = Forecast_idx(sales_df)

# Forecast Inventory for each product
Inventory_Forecast_df = Forecast_Inventory(sales_df)

# Save Forecast
save_forecasts(sales_forecast_df, Inventory_Forecast_df)

if __name__ == '__main__':
    forecast_and_save()

```



**FIGURE 5**



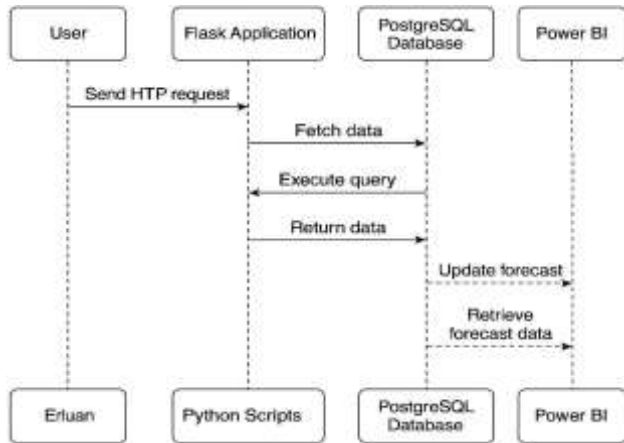


FIGURE 6

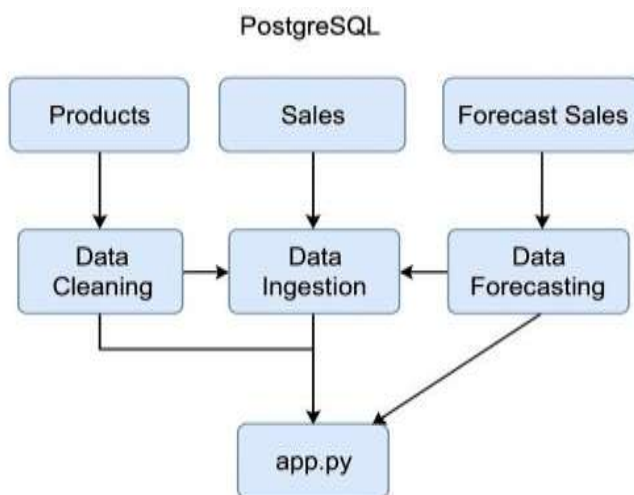


FIGURE 7

#### IV. RESULT AND DISCUSSION

The real-time inventory and sales dashboard offers precise and up-to-date data on the availability of products and ongoing sales. The purpose of this tool is to assist in making informed choices by providing a detailed and organized presentation of stock, sales patterns, and demand trends. Accurate monthly demand forecasting is facilitated by the use of index smoothing algorithms, which take into account seasonal patterns and fluctuations in customer behavior. Through rigorous data cleansing efforts, a significant enhancement in performance was achieved, resulting in a reduction of forecasting errors by as much as 98%. Additionally, data upload speeds were boosted by 80% after the implementation of optimized revenue processing scripts.

The dashboard, developed using power bi, features interactive and dynamic visualizations that allow users to quickly identify trends, anticipate shortages, and optimize restocking operations. By integrating postgresql for real-time data management, python for automation and prediction, and power bi for visualization, the

project offers a comprehensive solution for sales and inventory management. As a result, companies can now utilize automated ingestion, cleansing, analysis, and reporting in a unified platform. This system not only improves the accuracy of inventory planning through predictive analytics but also empowers decision-makers to make real-time decisions, thereby enhancing operational efficiency, minimizing the risk of overstocking or stockouts, and increasing overall supply chain agility.

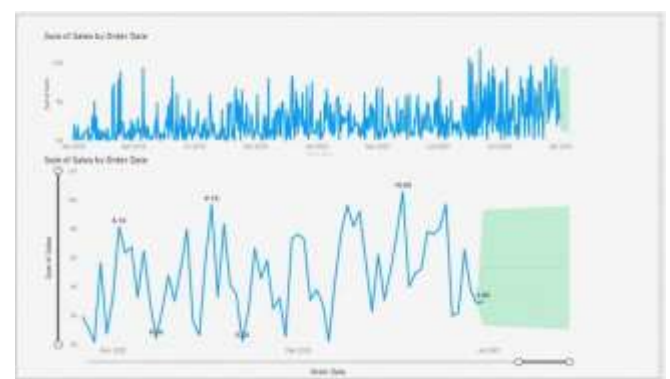


FIGURE 8



FIGURE 9

#### V. CONCLUSION

The present sales and inventory control dashboard is a remarkable combination of data engineering, predictive modeling, and real-time visualization, providing organizations with actionable insights. The process starts with a thorough data cleaning using Python, guaranteeing high-quality, consistent, and accurate data for subsequent tasks. After tidying up, the data is organized and optimized using pandas, and then stored in a PostgreSQL database, allowing for easy access and manipulation. The core forecasting engine is driven by a model that uses exponential smoothing to analyze historical sales data and forecast sales trends for the next 12 months. By anticipating future needs, businesses can effectively manage inventory levels, devise procurement plans, and optimize pricing and promotions.

The backend infrastructure, constructed using flask and sqlalchemy, provides a smooth user experience by enabling real-time data access, filtering, and report generation through a web interface. Power BI enhances business intelligence by establishing a direct connection to the postgresql database and providing dynamic, interactive dashboards. These visualizations improve user comprehension and aid in making informed choices at both operational and departmental levels.

This system enables organizations to shift from a reactive approach to inventory management, allowing them to anticipate and address shortages and overstock situations proactively. Future improvements will concentrate on integrating internet of things (iot)-based sensors for real-time inventory tracking and implementing advanced artificial intelligence (ai) models to enhance forecasting accuracy and business intelligence..

## VI. REFERENCES

- [1] R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, 3rd ed., OTexts, 2021.
- [2] PostgreSQL Global Development Group. (2024). PostgreSQL Documentation. [Online]. Available: <https://www.postgresql.org/docs/>
- [3] Flask. (2024). Flask Documentation. [Online]. Available: <https://flask.palletsprojects.com/>
- [4] Power BI. (2024). Microsoft Power BI Documentation. [Online]. Available: <https://learn.microsoft.com/en-us/power-bi/>
- [5] Wes McKinney, Python for Data Analysis, 2nd ed., O'Reilly Media, 201