Statistical Study & Data Analysis on

Queueing and Inventory Optimization in Retail: Insights from Transactional Data Analysis

Project work Submitted to
The Department of Statistics
Pondicherry University

By
Ms. Ananya Khanduri
Regd. No.: 23375007

As an Assignment for the
Partial Fulfilment of the Course work
STAT-537 Queueing And Inventory Theory

Under the Guidance of
Dr. P. Tirupathi Rao
Professor&Course Teacher

December - 2024

DR.TIRUPATHI RAO PADI

M.Sc., M.Phil., Ph.D., M.B.A., D.C.A., CP-S

PROFESSOR,Department of Statisti

Ramanujan School of Math. Sciences

PONDICHERRY UNIVERSITY

(A Central University)

R.V. Nagar, Kalapet,

Puducherry (UT) – 605014, India

Email: drtrpadi@pondiuni.ac.in;

traopadipu@gmail.com,

Phone: 9486492241 (Mobile),

9629862241 (WAN), 0413-2654-829(Office),

Certificate

This is to certifiy that Ms Ananya Khanduri has completed her project work as partial fulfilment fo the course work of Stat 537: Queueing And Inventory Theory submitted to the department fo Statistics, Pondicherry University in November 2024. She has carried out all the stages of project work right from the data collection to report making, on her own. The data that they have collected is from a real time context. No part of this work was carried out earlier in any format by any one for M.Sc./ MBA/ M.Tech/ etc. dissertation works or Ph.D. thesis.

Date: 5.12.2024 (P.Tirupathi Rao)

INDEX

Chapter 1:

Aim of the Project

This project is designed to analyze, model, and optimize the operations of a retail or business establishment by integrating **queueing theory** with **inventory management techniques**. Using transaction-level data, the goal is to develop a **hybrid model** that enables the business to balance **customer service efficiency** with **inventory cost-effectiveness**, leading to actionable insights and strategic recommendations.

1. Queueing Model Analysis

- Understanding Transaction Patterns: Using time-stamped sales data, the project investigates the arrival and service rates of transactions to understand the dynamic flow of customer activity.
- Model Selection: The project evaluates standard queueing models, such as:
    - M/M/1:(∞/FIFO)M/M/1: (\infty/FIFO): A single-server system with infinite queue capacity and first-come-first-serve order.
    - M/M/C:(N/FIFO)M/M/C: (N/FIFO): A multi-server system with finite queue capacity. The goal is to determine the most appropriate model based on system constraints, arrival rates (λ), and service rates (μ).
- Key Metrics: The queueing analysis calculates vital operational characteristics, including:
    - Probability of no sales (idle system): Understanding downtime.
    - Probability of at least one transaction: Assessing shop activity levels.
    - Expected idle time: Quantifying inefficiencies.
    - Expected waiting time: Estimating customer delay.
    - Number of completed transactions: Evaluating service throughput.

2. Inventory Management Optimization

- Demand Rate Estimation: By linking the transaction arrival rate (λ) to inventory demand (r), the project connects customer activity directly to stock requirements.
- Optimal Order Quantity: Using deterministic inventory models like the Economic Order Quantity (EOQ), the optimal replenishment quantity is calculated to minimize combined ordering and holding costs.
- Cycle Time Optimization: The analysis derives the average replenishment interval and the frequency of inventory cycles (n0) for maintaining stock levels efficiently.
- Cost Minimization: Total inventory cost is evaluated, including:
    - Holding costs (C1) per unit of stock.
    - Ordering costs (C0) per replenishment cycle. These insights enable cost-effective inventory planning.

3. Hybridization of Queueing and Inventory Models

- The integration bridges queueing dynamics with inventory planning:
    - Arrival Rate (λ): Serves as the foundation for both customer arrival patterns and inventory demand.
    - Service Rate (μ): Reflects the transaction efficiency, impacting customer satisfaction and sales throughput.
    - Loading Rate (k): Links inventory management to queueing behavior, allowing for synchronized operational decisions.
- This hybridization ensures that inventory replenishment aligns with real-time transaction activity, avoiding stockouts or overstocking.

4. Data-Driven Recommendations for the Shopkeeper

Based on the queueing and inventory analysis, the project provides actionable recommendations:

- Queue Management:
    - Optimize service rates to reduce waiting times and idle periods.
    - Adjust staffing or service operations during peak and off-peak periods to enhance efficiency.
- Inventory Control:
    - Plan replenishment schedules and quantities to minimize holding and ordering costs.
    - Implement strategies to align stock levels with demand patterns derived from queueing analysis.
- Overall Strategy**:**
    - Leverage transaction insights to design promotions, discounts, or other strategies during identified idle times.
    - Continuously monitor and adapt to changing transaction patterns for sustained optimization.

The aim of this project is to provide a comprehensive framework that integrates real-world sales transaction data into a cohesive operational model. By combining queueing theory with inventory management, the shopkeeper is equipped with:

- A deeper understanding of customer behavior and service dynamics.
- A robust inventory strategy that minimizes costs while ensuring product availability.
- Actionable insights to enhance overall business performance.

Dataset Structure:

- **Invoice ID**: A unique identifier for each transaction.
- **Branch**: Represents the branch of the store where the transaction occurred.
- **City**: Indicates the city where the store is located.
- **Customer type**: Specifies whether the customer is a member or a regular customer.
- **Product line**: The category or type of product purchased (e.g., electronics, clothing, etc.).
- **Unit price**: The price of a single unit of the product.
- **Quantity**: The number of units purchased in the transaction.
- **Tax 5%**: The calculated tax (5%) on the total price of the items purchased.
- **Total**: The total amount paid, including tax.
- **Date**: The date when the transaction occurred.
- **Time**: The time when the transaction occurred.
- **Payment**: The mode of payment used (e.g., cash, credit card, etc.).
- **cogs**: Cost of Goods Sold (COGS) - the direct cost attributed to the production of the goods sold.
- **gross margin percentage**: The gross profit margin expressed as a percentage.
- **gross income**: The gross profit made from the transaction (Total - COGS).
- **Rating**: Customer feedback rating, likely on a numeric scale.

Possible Objectives for Analysis:

1. **Sales Analysis**:
    - Identify trends in sales by branch, city, product line, or customer type.
    - Understand the performance of each product line in terms of revenue and quantity sold.
2. **Customer Behavior**:
    - Analyze the differences in purchasing patterns between member and regular customers.
    - Study the distribution of payment modes and how they vary across branches or cities.
3. **Time-Based Insights**:
    - Analyze sales trends by time of day, day of the week, or month.
    - Find peak transaction hours and dates for each branch or city.
4. **Tax and Revenue**:
    - Calculate the total tax collected for the study period.
    - Study the relationship between tax, total sales, and gross income.
5. **Profitability Analysis**:
    - Explore gross margins and gross income across different product lines.
    - Identify high-margin or low-margin products.
6. **Queueing and Inventory Modeling**:
    - Model customer arrivals and service rates using the time of transactions.
    - Estimate queue parameters such as waiting time, idle time, and service rate.
    - Calculate inventory-related metrics like optimal order quantity, cycle length, and total cost.

Suggestions for Analysis Workflow:

1. **Data Cleaning**:
    - Ensure all columns have consistent data formats (e.g., date and time fields).
    - Handle missing or outlier values, if any.
2. **Descriptive Analysis**:
    - Generate summary statistics for numerical variables like unit price, quantity, and gross income.
    - Create frequency distributions for categorical variables like branch, city, and payment mode.
3. **Visualization**:
    - Use plots (bar charts, histograms, and time series) to visualize trends and patterns.
    - Analyze the distribution of ratings to understand customer satisfaction.
4. **Queueing Model**:
    - Use transaction times to calculate inter-arrival and service rates.
    - Determine the best-fitting queueing model (e.g., M/M/1 or M/M/C).
5. **Inventory Optimization**:
    - Estimate demand rates from sales data.
    - Calculate key inventory metrics to optimize costs.

Chapter 2

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a critical phase in the data analysis process, playing a pivotal role in understanding the characteristics of a dataset before more in-depth analyses or modeling. Here are several reasons highlighting the importance of exploratory data analysis: Data Understanding: EDA helps in gaining a preliminary understanding of the dataset. It involves summarizing the main characteristics, identifying patterns, and detecting potential outliers.

Data Cleaning: By exploring the data, analysts can identify missing values, outliers, or inconsistencies. Addressing these issues during the EDA phase is crucial for improving the quality of the data.

Pattern Recognition: EDA allows the identification of patterns, trends, and relationships within the data. Recognizing these patterns is essential for formulating hypotheses and guiding subsequent analyses.

Visualization for Communication: Visualizations created during EDA serve as powerful tools for communicating findings to stakeholders. Graphs and charts can convey complex patterns more effectively than raw data.

Data Reduction: EDA can help in identifying redundant or irrelevant variables, leading to data reduction. This simplifies subsequent analyses and models, making them more interpretable.

Hypothesis Generation: By exploring the data, analysts can generate hypotheses about potential relationships or patterns. These hypotheses can then be tested rigorously in subsequent analyses.

Decision-Making Support: EDA provides valuable insights that support decision-making processes. Understanding the data thoroughly helps stakeholders make informed choices based on data-driven evidence.

Time and Cost Efficiency: Addressing data quality issues early in the analysis process, during EDA, can save time and resources. Detecting problems early prevents wasted efforts on faulty analyses or models.

In summary, exploratory data analysis is foundational for any data analysis or modeling project. It ensures that subsequent analyses are based on a solid understanding of the data, improving the reliability and validity of findings and conclusions.

Meaning and Scope of Descriptive Statistics:

Meaning: Descriptive statistics refers to the branch of statistics that involves the collection, organization, analysis, interpretation, and presentation of data. Its primary goal is to summarize and describe the main features of a dataset, providing a concise and understandable overview.

Scope: The scope of descriptive statistics includes:

Measures of Central Tendency: Describing the central or typical value of a dataset, such as the mean, median, or mode.

Measures of Dispersion: Indicating the spread or variability of data, including the range, variance, and standard deviation.

Frequency Distributions: Presenting the distribution of values and their frequencies.

Summary Statistics: Providing concise summaries, like percentiles, quartiles, and interquartile range.

Graphical Representations: Using charts and graphs to visually represent data distributions.

Need & Significance of Descriptive Statistics:

Need: Data Summarization: Descriptive statistics condenses large amounts of data into manageable summaries.

Pattern Recognition: It helps identify patterns, trends, and outliers within a dataset.

Data Exploration: Descriptive statistics serves as a preliminary step for in-depth data exploration. Communication: Summarized data is more easily communicable to a broad audience. Significance: Decision Making: Descriptive statistics aids in making informed decisions by providing a clear understanding of the data.

Comparisons: It facilitates comparisons between different groups or datasets. Problem Solving: Helps in identifying and addressing data-related issues.

Research: Essential in various fields for conducting preliminary analyses before more advanced statistical techniques are applied.

Statistical Tools used:

Some basic statistical tools and there definitions that are used in the project are mentioned below:

Mean:

- **Definition:** The mean, also known as the average, is the sum of all values in a dataset divided by the number of observations.
- **Formula:** Mean = (Sum of all values) / (Number of observations)

Median:

- **Definition:** The median is the middle value of a dataset when it is arranged in ascending or descending order. If there is an even number of observations, the median is the average of the two middle values.
- **Calculation:** Arrange data in order and find the middle value.

Mode:

- **Definition:** The mode is the value that appears most frequently in a dataset.
- **Calculation:** Identify the value(s) with the highest frequency.

Range:

- **Definition:** The range is the difference between the maximum and minimum values in a dataset.
- **Formula:** Range = (Maximum value) - (Minimum value)

Standard Deviation:

- **Definition:** The standard deviation measures the amount of variability or dispersion in a dataset.
- **Formula:** $\sigma = \sum (X - \mu) 2 n$

Variance:

Definition: Variance is the average of the squared differences from the mean. It provides a measure of the spread of values in a dataset.

Correlation:

- **Definition:** Correlation measures the strength and direction of a linear relationship between two variables. It ranges from -1 to 1, where -1 indicates a perfect negative correlation, 1 indicates a perfect positive correlation, and 0 indicates no correlation.

Regression:

- **Definition:** Regression analysis is used to model the relationship between a dependent variable and one or more independent variables. It helps predict the value of the dependent variable based on the values of the independent variables.

These are fundamental statistical tools that provide insights into the central tendency, variability, and relationships within a dataset. They form the basis for more advanced statistical analyses and are widely used in data analysis and research.

Graphs Constructed:

1. **Histogram:**
    - **Definition:** A histogram is a graphical representation of the distribution of a dataset. It consists of bars, where each bar represents a range of values (bin), and the height of the bar corresponds to the frequency or relative frequency of observations within that range.
2. **Bar Chart:**
    - **Definition:** A bar chart is a graph that presents categorical data with rectangular bars. The lengths of the bars are proportional to the values they represent, and the bars can be arranged vertically or horizontally.
3. **Pie Chart:**
    - **Definition:** A pie chart is a circular statistical graphic that is divided into slices to illustrate numerical proportions. Each slice represents a proportionate part of the whole, typically shown as a percentage.
4. **Line Chart:**
    - **Definition:** A line chart is a type of graph that displays information using a series of data points called 'markers' connected by straight line segments. It is often used to represent data over a continuous interval or time span.
5. **Scatter Plot:**
    - **Definition:** A scatter plot is a graph that shows the relationship between two continuous variables. Each point on the plot represents a pair of values, with one variable on the x-axis and the other on the y-axis.
6. **Box-and-Whisker Plot (Boxplot):**
    - **Definition:** A boxplot is a graphical summary of a distribution of data. It displays the minimum, first quartile, median, third quartile, and maximum of a dataset, helping to identify the central tendency and spread of the data.

1. Data Collection/Procurement in Excel

Objective: Collect and structure data that incorporates various types of measurement scales: **nominal**, **ordinal**, and **continuous**.

- Nominal Data**:** This includes categorical variables such as *City*, *Branch*, *Customer Type*, or *Product Line*. These variables represent categories without any inherent order.
- Ordinal Data: This includes variables with a meaningful order but without consistent intervals, such as *Rating* or *Customer Satisfaction Level* (e.g., "Poor," "Average," "Excellent").
- Continuous Data: This includes numerical variables measured on a continuous scale, such as *Unit Price*, *Quantity Sold*, *Total Amount*, or *Gross Income*.

2. Extract Data with a Timestamp

Objective: Use a dataset that includes a time stamp variable to track activities over time.

Steps:

1. Identify the column that represents date and time, such as *Date* or *Time*.
2. Ensure the timestamp is in a proper format (e.g., YYYY-MM-DD for dates and HH:MM:SS for times).
3. If the timestamp data is split into separate date and time columns, merge them into a single column using Excel formulas or a data manipulation tool.

3. Consider Data Related to Business/Trade Activity

Objective: Focus on data about everyday business/trade activities such as sales, inventory, or customer transactions.

- **Sales Data Example:**
  - Variables: Invoice ID, Branch, City, Product Line, Unit Price, Quantity, Total, Payment Method, etc.
  - Time-based variables: Date and Time of transactions.
  - Additional variables: Customer Type (new/regular), Discounts applied, Ratings, etc.
- **Inventory Data Example:**
  - Variables: Item ID, Category, Stock Levels, Restocking Date, Reorder Quantity, Supplier Information.
  - Time-based variables: Date of restocking, Expected delivery date.

4. Final Data Matrix in Excel

Objective: Create a structured dataset from the raw business activity data.

Steps to Arrive at the Final Data Matrix:

1. **Data Cleaning:**
   - Remove duplicates and handle missing data appropriately.
   - Standardize formats (e.g., consistent date-time formats, correct data types for numerical fields).
2. **Variable Classification:**
   - Categorize variables into **nominal**, **ordinal**, and **continuous**.

    - For example:
        - **Nominal:** Branch, City, Payment Method.
        - **Ordinal:** Ratings (e.g., 1–5 stars), Customer Type.
        - **Continuous:** Quantity, Unit Price, Total Sales, Tax.
3. **Merge and Restructure:**
    - Combine data from multiple sources (e.g., sales and inventory systems) if necessary.
    - Use tools like **Excel Pivot Tables** or **Power Query** to consolidate information.
    - Ensure each row represents a unique transaction or record with all relevant variables as columns.
4. **Enhance with Derived Variables:**
    - Add calculated fields such as:
        - Total Revenue = Quantity × Unit Price.
        - Gross Profit = Revenue - Cost of Goods Sold.
        - Time Elapsed Between Transactions = Difference in timestamps.
5. **Organize the Final Matrix:**
    - Ensure the data is organized in a tabular format:
        - Rows: Individual transactions or inventory records.
        - Columns: Variables such as *Invoice ID*, *Date/Time*, *Product Line*, *Total Revenue*, etc.

Outcome:

By following these steps, you ensure the dataset is:

- **Comprehensive:** Includes variables of interest with a variety of scales.
- **Time-Aware:** Captures the temporal nature of business activities.
- **Clean and Structured:** Ready for further analysis, such as queuing or inventory modeling.

Chapter 3: Queueing Model Analysis

3.1 Introduction to Queueing Theory

Queueing theory is a mathematical study of waiting lines, or queues. It provides tools for analyzing processes in which "customers" (which can be people, data packets, vehicles, etc.) arrive at a system, wait for service, and then depart after being served. Queueing theory is widely applied in operations research, computer science, telecommunications, manufacturing, and more.

Key Components of Queueing Theory

1. **Customers**: The entities that require service. These can be people, tasks, data packets, or any items that need processing.
2. **Servers**: The resources providing the service. Examples include checkout counters, network routers, or machines in a factory.
3. **Arrival Process**: Describes how customers arrive at the queue. This is often modeled using probability distributions, such as the Poisson process for random arrivals.
4. **Service Process**: Describes how customers are served. It is characterized by the service time, which might follow an exponential, deterministic, or general distribution.
5. **Queue Discipline**: Rules that determine the order in which customers are served. Common disciplines include:
    - First-In-First-Out (FIFO): Customers are served in the order of arrival.
    - Last-In-First-Out (LIFO): The most recent customer is served first.
    - Priority Queue: Customers with higher priority are served first.
6. **System Capacity**: The maximum number of customers that the system can accommodate. If the system is full, new arrivals may be blocked or lost.
7. **Population Source**: The source of customers, which can be finite or infinite.

Performance Metrics

Queueing theory focuses on evaluating key performance metrics such as:

- **Average waiting time**: The expected time a customer spends waiting.
- **Queue length**: The average number of customers in the queue.
- **System utilization**: The fraction of time the servers are busy.
- **Probability of system states**: For example, the likelihood that the system has no customers or that it is full.

3.2 Applications of Queueing Theory

- **Telecommunications**: Managing data packets in networks to reduce delays.
- **Healthcare**: Optimizing patient flow in hospitals.
- **Transportation**: Reducing congestion at toll booths and traffic lights.
- **Manufacturing**: Minimizing waiting time in production lines.
- **Retail**: Improving customer experience at checkout counters.

By studying these models and their characteristics, businesses and systems can optimize resource allocation, reduce bottlenecks, and enhance overall efficiency.

3.3 . Common Queuing Models

(M/M/1): (∞/FIFO)

- **Description:**
  - Single-server system with infinite capacity and First-In-First-Out (FIFO) queue discipline.
  - Poisson arrival process (random arrivals, rate λ\lambda).
  - Exponential service times (rate μ\mu).
- **Use Case:**
  Simple systems like single checkout counters or single-channel customer service.

(M/M/1): (N/FIFO)

- **Description:**
  - Single-server system with finite capacity (NN) and FIFO queue discipline.
  - Customers are turned away if the queue reaches capacity.
- **Use Case:**
  Scenarios with limited queue capacity, such as parking lots or call centers.

(M/M/C): (∞/FIFO)

- **Description:**
  - Multi-server system with infinite queue capacity and FIFO discipline.
  - Poisson arrival process and exponential service times.
- **Use Case:**
  Systems with multiple servers, like hospital emergency rooms or bank counters.

(M/M/C): (N/FIFO)

- **Description:**
  - Multi-server system with finite queue capacity and FIFO discipline.
  - Customers are rejected if the queue is full.
- **Use Case:**
  High-demand systems with limited resources, such as ticket counters or airport check-ins.

Identifying the Appropriate Queuing Model

Available Models:

- M/M/1: (∞/FIFO)
- M/M/1: (N/FIFO)
- M/M/C: (∞/FIFO)
- M/M/C: (N/FIFO)

Contextual Selection:
Based on the data, choose the model that best represents the scenario. For example:

- M/M/1: (∞/FIFO) is suitable for single-server systems with unlimited capacity and First-In-First-Out processing.

3.4 Exploring Queuing and Inventory Parameters

Queuing Parameters

- C: Number of service channels (servers).
- N: System capacity (maximum number of customers).
- λ\lambda: Arrival rate (customers per unit time).
- μ\mu: Service rate (transactions per unit time).

3.5 Market Basket Analysis and Identifying the Item with Maximum Sales

Market Basket Analysis involves analyzing sales transactions to identify patterns, such as which products are frequently purchased together or which products contribute the most to sales. Using tools like **Pivot Tables**, **Sorting**, and **Filtering** in Excel can help effectively analyze this data.

Steps to Perform Market Basket Analysis:

1. Preparing the Dataset

1. **Data Requirements:**
    - Ensure your dataset has the following columns: *Invoice ID*, *Branch*, *City*, *Product Line*, *Unit Price*, *Quantity Sold*, *Total Sales*, and *Payment Method*.
    - If the *Total Sales* column is missing, create it by multiplying *Unit Price* by *Quantity Sold* (e.g., =Unit Price * Quantity).
2. **Organize Data:**
    - Remove duplicates and clean missing data.
    - Format the columns appropriately (e.g., numerical values for sales data, text for product categories).

2. Creating a Pivot Table

A Pivot Table helps summarize large datasets by aggregating key metrics.

To Create a Pivot Table:

1. Select your data range and go to **Insert → Pivot Table**.
2. Place the Pivot Table in a new worksheet or within the same sheet.
3. Set up the following fields in the Pivot Table:
    - **Rows:** Product Line (or Item Name).
    - **Values:** Sum of Quantity Sold and Sum of Total Sales.
    - **Columns (Optional):** Branch or City (to compare performance across locations).

3. Sorting and Filtering

- **Sorting: Steps**
    - Sort the data in descending order based on *Total Sales* to identify the top-performing product or category.
    - For Quantity-based insights, sort by *Sum of Quantity Sold*.
- **Filtering:**
    - Use filters to focus on specific branches, cities, or time periods (e.g., filter by *Date* to analyze monthly trends).
    - If the dataset contains customer types, filter to compare performance across different segments (e.g., new vs. regular customers).

4. Adding a Calculated Field

If additional metrics are needed (e.g., profit margins), create a calculated field within the Pivot Table:

1. Go to **Pivot Table Analyze → Fields, Items & Sets → Calculated Field**.
2. Define the formula (e.g., Profit = Total Sales - Cost of Goods Sold).

Arriving at Summary Data

After setting up the Pivot Table and applying sorting and filtering tools:

1. Review the aggregated data for each *Product Line* or item.
2. Summarize the key metrics:
    - Total Quantity Sold.
    - Total Revenue (Sum of Total Sales).
    - Highest-performing branches or cities for each product.

Summary and Insights

1. **Top Performing Product Line:**
   From the table above, *Electronics* has the highest total sales of $67,500, making it the most profitable product line.
2. **Quantity vs. Revenue Analysis:**
    - *Groceries* have the highest quantity sold but generate less revenue due to lower unit prices.
    - *Fashion* generates higher revenue relative to quantity sold, suggesting a premium pricing strategy.
3. **Branch or City Insights (Optional):**
    - By adding *Branch* or *City* as a column in the Pivot Table, you can identify where specific products perform the best, helping in regional targeting.

Key Takeaways

Using Pivot Tables, sorting, and filtering tools provides actionable insights for Market Basket Analysis. In this case, the most successful product line (*Electronics*) can be prioritized for marketing and inventory management to drive business growth.

3.6 Analyzing Transactional Data with Time-Based Variables

This report provides a detailed breakdown of how to analyze transactional data by separating time-related variables, extracting specific time units, and calculating inter-transaction intervals. The process is organized into the following steps:

Separating Specific Variable Data

The raw dataset includes a *Time Stamp* variable, which records the date and time of each transaction. To focus on analyzing time-related patterns:

- Extract the *Time Stamp* data as a separate column for analysis.
- Ensure the format is consistent (e.g., YYYY-MM-DD HH:MM:SS).
- Identify transactions that occur at different times throughout the study period to capture temporal patterns such as peak transaction hours or off-peak trends.

Extracting Year, Month, Week, and Day Variables

The *Time Stamp* variable can be decomposed into its constituent units for temporal analysis:

1. **Year**: Use the first four digits of the time stamp to group transactions by year.
2. **Month**: Extract the month (MM) from the time stamp. Add both the month number (e.g., 01 for January) and its name (e.g., January) for clarity.
3. **Week**: Convert the *Time Stamp* to its corresponding week of the year using a week numbering function.
4. **Day**: Extract the day (DD) or the day of the week (e.g., Monday, Tuesday).

In Excel, the following functions can help:

- **Year**: =YEAR(TimeStamp)
- **Month Number**: =MONTH(TimeStamp)
- **Month Name**: =TEXT(TimeStamp, "mmmm")
- **Week Number**: =WEEKNUM(TimeStamp)
- **Day Name**: =TEXT(TimeStamp, "dddd")

Exploring Units of Time (Hours, Minutes, Seconds)

To analyze finer-grained time intervals:

- **Hour**: Extract the hour portion (HH) of the time stamp to identify peak transaction times.
- **Minutes**: Extract the minute portion (MM) for a more detailed breakdown.
- **Seconds**: Extract the seconds (SS) to measure precise intervals.

Use the following Excel functions:

- **Hour**: =HOUR(TimeStamp)
- **Minute**: =MINUTE(TimeStamp)
- **Second**: =SECOND(TimeStamp)

Total Seconds Calculation:

1. Convert each transaction time into total seconds since the start of the day using:
   =HOUR(TimeStamp)*3600 + MINUTE(TimeStamp)*60 + SECOND(TimeStamp)
2. To convert total seconds into minutes, divide the value by 60:
   =TotalSeconds/60

Calculating Inter-Transaction Time

To measure the time difference between consecutive transactions:

1. **Sort the Transactions**: Ensure transactions are sorted in chronological order based on the *Time Stamp*.
2. **Calculate Differences**: Subtract the time stamp of the previous transaction from the current one.
   - In Excel, use: =TimeStamp(CurrentRow) - TimeStamp(PreviousRow).
   - Format the result to display the difference in days, hours, minutes, or seconds as required.

Steps in Detail:

- Convert the time difference into seconds:
  =(TimeStamp(CurrentRow) - TimeStamp(PreviousRow))*86400
  (Multiply by 86400 because there are 86,400 seconds in a day).
- Alternatively, to calculate the inter-transaction time in minutes:
  =(TimeStamp(CurrentRow) - TimeStamp(PreviousRow))*1440
  (Multiply by 1440 because there are 1,440 minutes in a day).

Insights and Applications

1. **Peak Hours Analysis**: By grouping transactions by hour, you can identify when business activity is at its highest.
2. **Daily and Weekly Trends**: Analyzing transactions by day and week reveals patterns such as which days of the week have the most sales.
3. **Inter-Transaction Intervals**: Understanding the time gaps between transactions helps optimize resource allocation (e.g., staffing during high-frequency periods).

Comprehensive Analysis Report

This report focuses on analyzing the transactional dataset for queuing and inventory management insights. Key steps, mathematical relations, and contextual interpretations are outlined as follows:

---

11. Average Inter-Transaction Time and Service Rate Calculation

Objective:

- Calculate the average time between transactions for the session or day under study.

Steps:

1. Use the inter-transaction times to find the average:

   Mean Time(T¯)=∑TN\text{Mean Time} (\bar{T}) = \frac{\sum T}{N}

   where TT is the inter-transaction time and NN is the total number of transactions.

2. As inter-service times follow an exponential distribution, the service rate (μ\mu) is:

   μ=1Mean Time\mu = \frac{1}{\text{Mean Time}}

Contextual Insight:
The service rate (μ) indicates how quickly transactions are processed on average.

R codes For Queueing Model Analysis

```
# Load necessary library
library(readxl)  # To read Excel file

# Step 1: Load the data
library(readxl)
dataset_queueing <- read_excel("dataset queueing.xlsx")
View(dataset_queueing)

# Step 2: Preprocess the data
# Extract necessary columns and group by Invoice ID
transactions_data <- aggregate(`Product line` ~ `Invoice ID`,
dataset_queueing, paste, collapse = ",")
transactions_data
```

	Invoice ID	Product line
1	101-17-6199	Electronic accessories
2	101-81-4070	Electronic accessories
3	102-06-2002	Fashion accessories
4	102-77-2261	Health and beauty
5	105-10-6182	Sports and travel
6	105-31-1824	Sports and travel
7	106-35-6779	Fashion accessories
8	109-28-2512	Fashion accessories
9	109-86-4363	Health and beauty
10	110-05-6330	Sports and travel
11	110-48-7033	Food and beverages
12	114-35-5271	Electronic accessories
13	115-38-7388	Electronic accessories
14	115-99-4379	Food and beverages
15	118-62-1812	Electronic accessories
16	120-06-4233	Electronic accessories
17	120-54-2248	Fashion accessories
18	122-61-9553	Fashion accessories
19	123-19-1176	Home and lifestyle
20	123-35-4896	Food and beverages
21	124-31-1458	Electronic accessories
22	125-45-2293	Home and lifestyle
23	126-54-1082	Home and lifestyle
24	127-47-6963	Health and beauty
25	129-29-8530	Sports and travel
26	130-67-4723	Food and beverages
27	130-98-8941	Fashion accessories
28	131-15-8856	Home and lifestyle
29	131-70-8179	Fashion accessories
30	132-23-6451	Health and beauty
31	132-32-9879	Electronic accessories
32	133-14-7229	Health and beauty
33	133-77-3154	Home and lifestyle
34	134-54-4720	Electronic accessories
35	134-75-2619	Electronic accessories
36	135-13-8269	Fashion accessories
37	135-84-8019	Electronic accessories
38	136-08-6195	Food and beverages
39	137-63-5492	Fashion accessories
40	137-74-8729	Food and beverages
41	138-17-5109	Food and beverages
42	139-20-0155	Electronic accessories

```
43  139-32-4183      Health and beauty
44  139-52-2867     Food and beverages
45  142-63-6033      Health and beauty
46  142-72-4741     Food and beverages
47  144-51-6085     Home and lifestyle
48  145-94-9061      Sports and travel
49  146-09-5432     Food and beverages
50  148-41-7930      Health and beauty
51  148-82-2527     Home and lifestyle
52  149-14-0304      Health and beauty
53  149-15-7606      Sports and travel
54  149-61-1929     Home and lifestyle
55  149-71-6266     Home and lifestyle
56  150-89-8043     Food and beverages
57  151-16-1484 Electronic accessories
58  151-27-8496     Home and lifestyle
59  151-33-7434     Food and beverages
60  152-03-4217      Sports and travel
```

Step 3: Convert to transaction format

```
>transactions_list <- strsplit(transactions_data$`Product line`, ",")
>transactions_list
[[960]]
[1] "Food and beverages"

[[961]]
[1] "Home and lifestyle"

[[962]]
[1] "Home and lifestyle"

[[963]]
[1] "Fashion accessories"

[[964]]
[1] "Home and lifestyle"

[[965]]
[1] "Electronic accessories"

[[966]]
[1] "Sports and travel"

[[967]]
[1] "Sports and travel"

[[968]]
[1] "Food and beverages"

[[969]]
[1] "Electronic accessories"

[[970]]
[1] "Health and beauty"

[[971]]
[1] "Home and lifestyle"
```

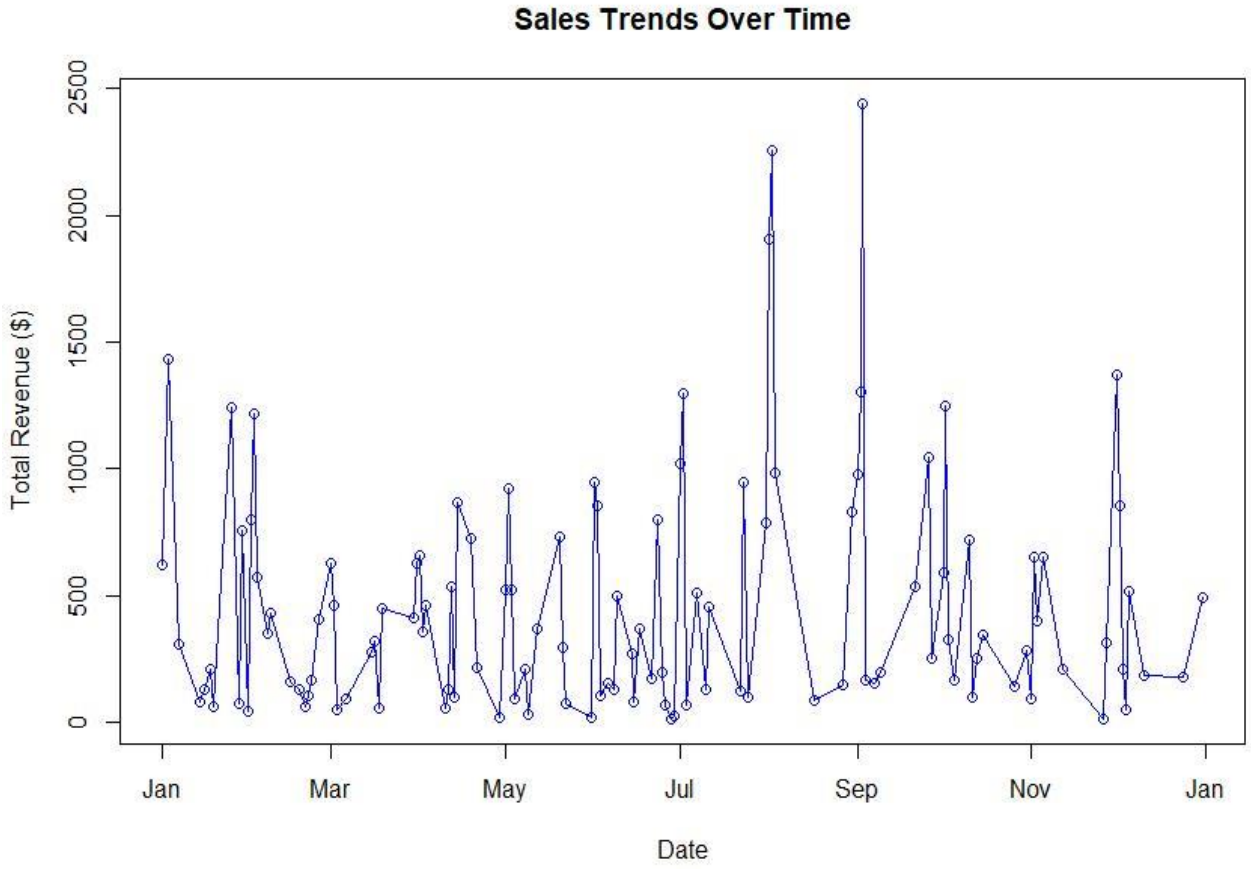
```
[[972]]
[1] "Sports and travel"

[[973]]
[1] "Health and beauty"

[[974]]
[1] "Sports and travel"

[[975]]
[1] "Health and beauty"

[[976]]
[1] "Electronic accessories"

[[977]]
[1] "Sports and travel"

[[978]]
[1] "Sports and travel"

[[979]]
[1] "Sports and travel"

[[980]]
[1] "Fashion accessories"

[[981]]
[1] "Health and beauty"

[[982]]
[1] "Sports and travel"

[[983]]
[1] "Fashion accessories"

[[984]]
[1] "Food and beverages"

[[985]]
[1] "Health and beauty"

[[986]]
[1] "Fashion accessories"

[[987]]
[1] "Electronic accessories"

[[988]]
[1] "Food and beverages"

[[989]]
[1] "Home and lifestyle"

[[990]]
[1] "Sports and travel"

[[991]]
[1] "Health and beauty"
```

```
[[992]]
[1] "Food and beverages"

[[993]]
[1] "Electronic accessories"

[[994]]
[1] "Sports and travel"

[[995]]
[1] "Home and lifestyle"

[[996]]
[1] "Sports and travel"

[[997]]
[1] "Home and lifestyle"

[[998]]
[1] "Food and beverages"

[[999]]
[1] "Electronic accessories"

[[1000]]
[1] "Electronic accessories"
```

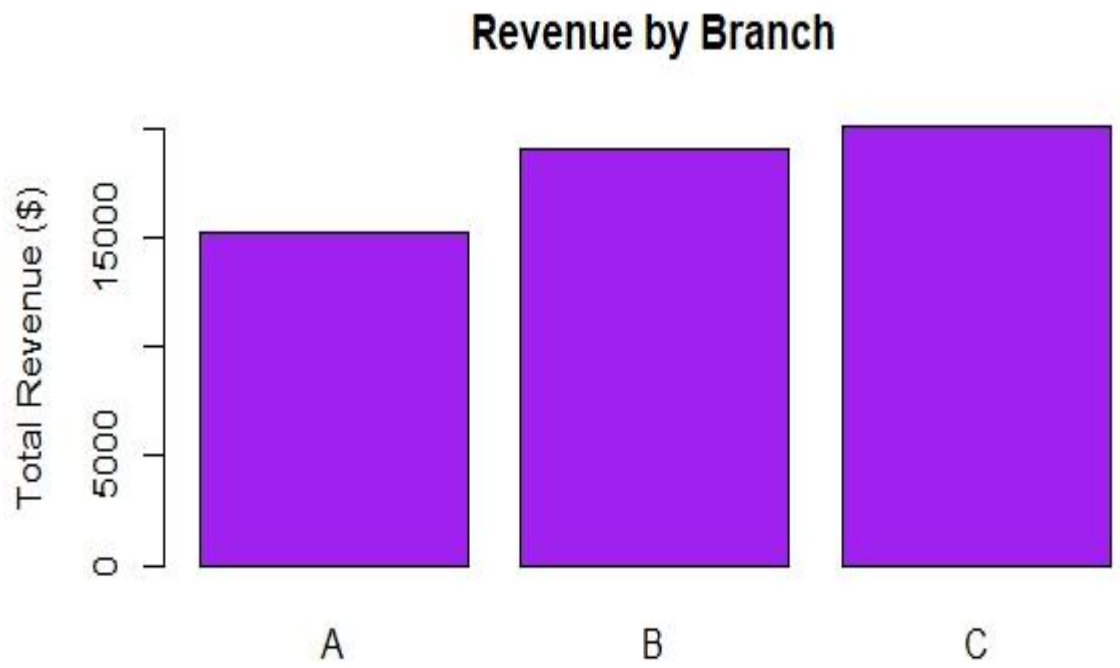
```
> unique_items <- unique(unlist(transactions_list))
> unique_items
[1] "Electronic accessories" "Fashion accessories"
[3] "Health and beauty"      "Sports and travel"
[5] "Food and beverages"     "Home and lifestyle"
```

```
# Create a binary transaction matrix
> transaction_matrix <- sapply(unique_items, function(item) {
 sapply(transactions_list, function(transaction) item %in%
transaction)
})
> transaction_matrix <- as.data.frame(transaction_matrix)
> transaction_matrix
Electronic accessories Fashion accessories
1                    TRUE                FALSE
2                    TRUE                FALSE
3                   FALSE                 TRUE
4                   FALSE                FALSE
5                   FALSE                FALSE
6                   FALSE                FALSE
7                   FALSE                 TRUE
8                   FALSE                 TRUE
9                   FALSE                FALSE
10                  FALSE                FALSE
Health and beauty Sports and travel Food and beverages
1               FALSE             FALSE               FALSE
2               FALSE             FALSE               FALSE
3               FALSE             FALSE               FALSE
4                TRUE             FALSE               FALSE
5               FALSE              TRUE               FALSE
```

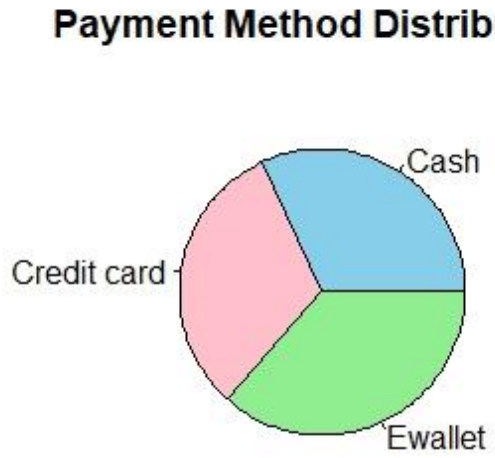
```
6                FALSE              TRUE              FALSE
7                FALSE             FALSE              FALSE
8                FALSE             FALSE              FALSE
9                 TRUE             FALSE              FALSE
10               FALSE              TRUE              FALSE
Home and lifestyle
1                 FALSE
2                 FALSE
3                 FALSE
4                 FALSE
5                 FALSE
6                 FALSE
7                 FALSE
8                 FALSE
9                 FALSE
10                FALSE
# Step 4: Generate summary
# Calculate item frequency
> item_frequency <- colSums(transaction_matrix)
> item_frequency <- sort(item_frequency, decreasing = TRUE)
Fashion accessories     Food and beverages
                  178                    174
Electronic accessories      Sports and travel
                  170                    166
    Home and lifestyle      Health and beauty
                  160                    152


# 1. Sales trends over time
> windows(width = 10, height = 7)
> sales_trend <- aggregate(Total ~ Date, data = data, sum)
> plot(sales_trend$Date, sales_trend$Total, type = "o", col = "blue",
+      main = "Sales Trends Over Time", xlab = "Date", ylab = "Total
Revenue ($)")
```




```
# Total revenue by branch
> par(mar = c(5, 5, 4, 2))
> revenue_by_branch <- aggregate(Total ~ Branch, data = data, sum)
> barplot(revenue_by_branch$Total, names.arg =
revenue_by_branch$Branch,
+         col = "purple", main = "Revenue by Branch", ylab = "Total
Revenue ($)")
```




```
#Distribution of payment methods
> par(mar = c(4, 4, 4, 4))
> payment_distribution <- table(data$Payment)
> pie(payment_distribution, col = c("skyblue", "pink", "lightgreen"),
    +     main = "Payment Method Distribution")
```




```
#Customer ratings distribution
> par(mar = c(5, 5, 4, 2))
> hist(data$Rating, breaks = 10, col = "orange",
+      main = "Customer Ratings Distribution", xlab = "Rating", ylab
= "Frequency")
```


```
# Print top 3 items
> # Print top 3 items
> cat("Top 3 items:\n")
Top 3 items:
> print(head(item_frequency, 3))
```

```
  Fashion accessories     Food and beverages
                  178                    174
Electronic accessories
                  170
> # Step 5: Display association (manual inspection)
> # Example: Checking co-occurrence of top 2 items
> top_items <- names(head(item_frequency, 2))
> co_occurrence <- rowSums(transaction_matrix[, top_items])
> cat("Co-occurrence of", top_items[1], "and", top_items[2], ":\n")
Co-occurrence of Fashion accessories and Food and beverages :
> print(sum(co_occurrence == 2))
[1] 0
> # Step 6: Save results
> write.csv(item_frequency, "item_frequency.csv", row.names = FALSE)



> # Load the data
> library(readxl)
> data <- read_excel("fashion.xlsx")
> View(data)
> # Ensure Date is in the correct format
> data$Date <- as.Date(data$Date, format = "%Y-%m-%d")
> # Ensure Time is in HH:MM:SS format; check if Time needs formatting
> data$Time <- format(hms::as_hms(data$Time), "%H:%M:%S")
> # Combine Date and Time into a single POSIXct DateTime object
> data$DateTime <- as.POSIXct(paste(data$Date, data$Time), format =
"%Y-%m-%d %H:%M:%S")
> data$DateTime
 [1] "2023-01-01 13:55:00 IST" "2023-01-03 13:22:00 IST"
 [3] "2023-01-03 19:49:00 IST" "2023-01-03 16:23:00 IST"
 [5] "2023-02-01 18:09:00 IST" "2023-02-01 13:40:00 IST"
 [7] "2023-02-02 13:23:00 IST" "2023-02-02 18:31:00 IST"
 [9] "2023-02-02 20:42:00 IST" "2023-02-02 14:05:00 IST"
[11] "2023-02-03 13:01:00 IST" "2023-02-03 16:35:00 IST"
[13] "2023-03-02 15:59:00 IST" "2023-03-02 20:13:00 IST"
[15] "2023-03-03 20:06:00 IST" "2023-04-01 19:01:00 IST"
[17] "2023-04-02 19:38:00 IST" "2023-04-02 15:44:00 IST"
[19] "2023-04-03 11:27:00 IST" "2023-05-01 20:54:00 IST"
[21] "2023-05-01 12:40:00 IST" "2023-05-01 11:32:00 IST"
[23] "2023-05-01 20:30:00 IST" "2023-05-02 19:05:00 IST"
[25] "2023-05-02 14:14:00 IST" "2023-05-03 12:29:00 IST"
[27] "2023-05-03 17:07:00 IST" "2023-05-03 17:43:00 IST"
[29] "2023-06-01 12:12:00 IST" "2023-06-01 11:51:00 IST"
[31] "2023-06-02 18:07:00 IST" "2023-06-02 17:20:00 IST"
[33] "2023-06-03 11:55:00 IST" "2023-07-01 15:01:00 IST"
[35] "2023-07-01 20:52:00 IST" "2023-07-02 17:55:00 IST"
[37] "2023-07-02 12:23:00 IST" "2023-07-02 17:59:00 IST"
[39] "2023-07-02 19:30:00 IST" "2023-07-03 14:53:00 IST"
[41] "2023-08-01 16:17:00 IST" "2023-08-01 14:11:00 IST"
[43] "2023-08-01 20:08:00 IST" "2023-08-01 19:07:00 IST"
[45] "2023-08-02 15:31:00 IST" "2023-08-02 16:20:00 IST"
[47] "2023-08-02 13:00:00 IST" "2023-08-02 11:39:00 IST"
[49] "2023-08-03 13:21:00 IST" "2023-08-03 20:01:00 IST"
[51] "2023-09-01 14:20:00 IST" "2023-09-01 13:46:00 IST"
[53] "2023-09-01 13:34:00 IST" "2023-09-02 10:00:00 IST"
[55] "2023-09-02 20:31:00 IST" "2023-09-03 14:57:00 IST"
[57] "2023-09-03 15:36:00 IST" "2023-09-03 14:36:00 IST"
[59] "2023-09-03 13:18:00 IST" "2023-10-01 10:33:00 IST"
```

```
 [61] "2023-10-02 12:28:00 IST" "2023-10-02 12:56:00 IST"
 [63] "2023-10-02 13:56:00 IST" "2023-10-03 12:17:00 IST"
 [65] "2023-10-03 17:38:00 IST" "2023-11-01 11:20:00 IST"
 [67] "2023-11-02 16:19:00 IST" "2023-11-03 12:04:00 IST"
 [69] "2023-12-01 15:48:00 IST" "2023-12-01 16:23:00 IST"
 [71] "2023-12-01 11:25:00 IST" "2023-12-02 12:58:00 IST"
 [73] "2023-12-02 17:49:00 IST" "2023-01-07 17:16:00 IST"
 [75] "2023-01-14 19:30:00 IST" "2023-01-16 14:55:00 IST"


# Extract Year, Month (numeric and name), Week, and Day
> data$Year <- format(data$DateTime, "%Y")
> data$Month <- format(data$DateTime, "%m")  # Numeric month
> data$Month_Name <- format(data$DateTime, "%B")  # Full month name
> data$Week <- format(data$DateTime, "%U")  # Week number
> data$Day <- format(data$DateTime, "%d")  # Day of the month
> data$Day_Name <- format(data$DateTime, "%A")  # Day name
> # Task 2: Extract Hours, Minutes, Seconds and calculate total
seconds
> data$Hour <- format(data$DateTime, "%H")
> data$Minute <- format(data$DateTime, "%M")
> data$Second <- format(data$DateTime, "%S")
> # Calculate total seconds since the start of the day
> data$Total_Seconds <- as.numeric(data$Hour) * 3600 +
as.numeric(data$Minute) * 60 + as.numeric(data$Second)
> data$Total_Seconds
  [1] 50100 48120 71340 58980 65340 49200 48180 66660 74520
 [10] 50700 46860 59700 57540 72780 72360 68460 70680 56640
 [19] 41220 75240 45600 41520 73800 68700 51240 44940 61620
 [28] 63780 43920 42660 65220 62400 42900 54060 75120 64500
 [37] 44580 64740 70200 53580 58620 51060 72480 68820 55860
 [46] 58800 46800 41940 48060 72060 51600 49560 48840 36000
 [55] 73860 53820 56160 52560 47880 37980 44880 46560 50160
 [64] 44220 63480 40800 58740 43440 56880 58980 41100 46680
 [73] 64140 62160 70200 53700 42480 63960 46320 58200 64980
 [82] 70200 63360 44520 68220 49260 67020 52920 69720 55560
 [91] 71280 56700 46020 72180 68520 40020 40560 36660 54240
[100] 66000 59160 58980 55980 47460 68760 57120 74220 64080
[109] 55740 61680 56940 70920 55380 59280 50640 46920 44400
[118] 40500 71700 65040 42300 52920 56760 47880 69600 68340
[127] 70740 48180 66600 39900 65640 70080 53460 37380 67980
[136] 36360 65940 45780 69600 53700 67020 65100 71040 69840
[145] 38580 49260 42960 57960 51120 44700 46260 71280 38160
[154] 37980 36000 39000 61140 54360 45840 53580 45060 55260
[163] 63000 56520 65040 39120 48840 70740 48480 36720 45480
[172] 66660 38580 65940 38940 66900 71820 70740

# Convert total seconds to minutes
> data$Total_Minutes <- data$Total_Seconds / 60
> data$Total_Minutes
  [1]  835  802 1189  983 1089  820  803 1111 1242  845  781
 [12]  995  959 1213 1206 1141 1178  944  687 1254  760  692
 [23] 1230 1145  854  749 1027 1063  732  711 1087 1040  715
 [34]  901 1252 1075  743 1079 1170  893  977  851 1208 1147
 [45]  931  980  780  699  801 1201  860  826  814  600 1231
 [56]  897  936  876  798  633  748  776  836  737 1058  680
 [67]  979  724  948  983  685  778 1069 1036 1170  895  708
 [78] 1066  772  970 1083 1170 1056  742 1137  821 1117  882
 [89] 1162  926 1188  945  767 1203 1142  667  676  611  904
[100] 1100  986  983  933  791 1146  952 1237 1068  929 1028
```

```
[111]  949 1182  923  988  844  782  740  675 1195 1084  705
[122]  882  946  798 1160 1139 1179  803 1110  665 1094 1168
[133]  891  623 1133  606 1099  763 1160  895 1117 1085 1184
[144] 1164  643  821  716  966  852  745  771 1188  636  633
[155]  600  650 1019  906  764  893  751  921 1050  942 1084
[166]  652  814 1179  808  612  758 1111  643 1099  649 1115
[177] 1197 1179
```

Task 3: Find the inter-transaction time between consecutive transactions

```
> # Sort data by DateTime
> data <- data[order(data$DateTime), ]
> data
`Invoice ID` Branch City    `Customer type` `Product line`
   <chr>        <chr>  <chr>   <chr>           <chr>
 1 651-88-7328  A      Yangon  Normal          Fashion acces~
 2 189-98-2939  C      Naypyi~ Normal          Fashion acces~
 3 247-11-2470  A      Yangon  Member          Fashion acces~
 4 339-96-8318  B      Mandal~ Member          Fashion acces~
 5 460-93-5834  A      Yangon  Normal          Fashion acces~
 6 868-52-7573  B      Mandal~ Normal          Fashion acces~
 7 320-85-2052  B      Mandal~ Normal          Fashion acces~
 8 377-79-7592  C      Naypyi~ Member          Fashion acces~
 9 272-65-1806  A      Yangon  Normal          Fashion acces~
10 660-29-7083  C      Naypyi~ Normal          Fashion acces~

# Calculate inter-transaction time in seconds
> data$Inter_Transaction_Time <- c(NA,
diff(as.numeric(data$DateTime)))
> data$Inter_Transaction_Time
  [1]      NA  170820   10860   12360  336420  612840  156300
  [8]  161580  107880  500760   11880  265980   84780    6840
 [15]  147120   91080   16140    2880   66360    2520   15960
 [22]    7860   58740   12840  335160   17760   72300  621600
 [29]  245040   15720  158220   75720  112560  255540  317100
 [36]  103920   15240   85980  227400  773700  103980   11760
 [43]  165960   86220  947400   77880  107400     300   74280
 [50]   14040   56940  620700  103500   76260   78060   92340
 [57]  427260  186780  675660  158940    4080   28200    1440
 [64]   62400    8040    9420   62640   16680    2160   73260
 [71]  341880   83880  255300   31200  684540   63660   97020
 [78]  781440   72300    1260    3960  100920    2820    4380
 [85]   59700  284640  175200   63840  450420   59700  198540
 [92]  350040  140100   16080  187320   54780  202380   66240
 [99]  181080   15540    5520   55860   19920     240    5460
[106]   69780  345720  272520   84480    5940  949200   55140
[113]  183480  512100   94500    6900     660   10200    3660
[120]   55860    4860    4320    4740    2940   72300    3360
[127]   24000 1183800  889020  226080  183480     720    2040
[134]   70800    1980   35880   48540   11880    4680    1260
[141]    2340   69240  281340  166020 1028280  439740   77880
[148]  338520   93300    1680    3600   80460   19260  164580
[155]  439740   79920  181320  146880  960120  367500  142860
[162]  104340   71100  177840  593040 1218360  107580  320040
[169]   15780    2100   74100   17460   60840  113760   59400
[176]  459960 1214520  603720
```

```
# Task: Calculate the average inter-transaction time for the session or day
> avg_inter_time <- mean(data$Inter_Transaction_Time, na.rm = TRUE)
> avg_inter_time
[1] 177798
> service_rate <- 1 / avg_inter_time  # Service rate (lambda)
> service_rate
[1] 5.624361e-06
> avg_inter_time <- mean(data$Inter_Transaction_Time/60, na.rm = TRUE)
> avg_inter_time
[1] 2963.299
> service_rate <- 1 / avg_inter_time  # Service rate (lambda)
> service_rate
[1] 0.0003374617

# Display results
> cat("Average Inter-Transaction Time (minutes):", avg_inter_time, "\n")
Average Inter-Transaction Time (minutes): 2963.299
> cat("Service Rate (1/Mean Time):", service_rate, "\n")
Service Rate (1/Mean Time): 0.0003374617


# Task: Set service rate as demand rate (r = lambda)
> demand_rate <- service_rate
> cat("Demand Rate (r):", demand_rate, "\n")
Demand Rate (r): 0.0003374617
> # Task 12: Set service rate as demand rate (r = lambda)
> demand_rate <- service_rate
> cat("Demand Rate (r):", demand_rate, "\n")
Demand Rate (r): 0.0003374617


> # Task : Calculate inventory loading rate (k = lambda)
> inventory_loading_rate <- service_rate  # k = lambda
> cat("Inventory Loading Rate (k):", inventory_loading_rate, "\n")

Inventory Loading Rate (k): 0.0003374617


# Task : Explore queueing and inventory parameters
> # Queueing parameters
> queue_params <- list(
```

```
+   lambda = service_rate,  # Arrival rate
+   mu = service_rate * 1.2,  # Assuming service rate is 20% faster
than arrival rate
+   C = 1,  # Single server
+   N = nrow(data)  # Total number of transactions
+ )
> # Inventory parameters
> inventory_params <- list(
+   r = demand_rate,  # Demand rate
+   k = inventory_loading_rate,  # Loading rate
+   t = avg_inter_time,  # Average inter-arrival time
+   n = length(unique(data$Date)),  # Number of days in the dataset
+   q = 50,  # Example reorder quantity (to be adjusted based on
context)
+   D = sum(data$Quantity, na.rm = TRUE),  # Total demand
+   C1 = 2,  # Example holding cost per unit
+   C0 = 100  # Example ordering cost
+ )
> # Display queueing and inventory parameters
> cat("Queueing Parameters:\n")
Queueing Parameters:
> print(queue_params)
$lambda
[1] 0.0003374617

$mu
[1] 0.000404954

$C
[1] 1

$N
[1] 178


# Task : Identify the appropriate queueing model
> if (queue_params$C == 1 && is.infinite(queue_params$N)) {
+   queue_model <- "M/M/1: (∞/FIFO)"
+ } else if (queue_params$C == 1 && !is.infinite(queue_params$N)) {
+   queue_model <- "M/M/1: (N/FIFO)"
+ } else if (queue_params$C > 1 && is.infinite(queue_params$N)) {
+   queue_model <- "M/M/C: (∞/FIFO)"
+ } else {
+   queue_model <- "M/M/C: (N/FIFO)"
+ }
> cat("Selected Queueing Model:", queue_model, "\n")
Selected Queueing Model: M/M/1: (N/FIFO)


# Task : Estimate queue characteristics
> prob_no_sales <- exp(-queue_params$lambda / queue_params$mu)  #
P(0)
```

```
> prob_at_least_one_sale <- 1 - prob_no_sales  # P(at least 1)
> idle_time <- 1 / queue_params$mu  # Expected idle time
> completed_sales <- queue_params$lambda * avg_inter_time  # Expected
number of sales
> waiting_time <- 1 / (queue_params$mu - queue_params$lambda)  #
Expected waiting time
> # Display queue characteristics
> cat("Probability of no sales:", prob_no_sales, "\n")
Probability of no sales: 0.4345982
> cat("Probability of at least one sale:", prob_at_least_one_sale,
"\n")
Probability of at least one sale: 0.5654018
> cat("Expected idle time:", idle_time, "\n")
Expected idle time: 2469.416
> cat("Expected number of completed sales:", completed_sales, "\n")
Expected number of completed sales: 1
> cat("Expected waiting time:", waiting_time, "\n")
Expected waiting time: 14816.5
```

Chapter 4: Queueing Model Analysis

4.1 Introduction to Inventory Theory:

Inventory theory is the branch of operations research and management science that focuses on the management of inventories or stocks. It provides mathematical models and strategies for managing inventory levels efficiently to balance costs and ensure smooth operations in various settings, such as manufacturing, retail, and logistics.

Key Objectives of Inventory Theory

1. **Minimize Costs**: Inventory management aims to reduce costs associated with holding, ordering, and stockouts.
2. **Meet Demand**: Ensure sufficient inventory to satisfy customer or production demand without excessive delays.
3. **Optimize Resource Allocation**: Determine when and how much inventory to order or produce.

4.2 Types of Inventory

1. **Raw Materials**: Items used in the production process.
2. **Work-In-Progress (WIP)**: Partially completed products.
3. **Finished Goods**: Completed products ready for sale or distribution.
4. **Safety Stock**: Extra inventory kept as a buffer against uncertainty.
5. **Pipeline Inventory**: Stock in transit between locations.

4.3 Key Costs in Inventory Management

1. **Holding Costs**: Costs associated with storing inventory, such as warehousing, insurance, and depreciation.
2. **Ordering Costs**: Costs incurred when replenishing inventory, including administrative and transportation costs.
3. **Shortage Costs**: Costs arising from stockouts, such as lost sales, production delays, or customer dissatisfaction.
4. **Purchase Costs**: Cost of acquiring inventory, often influenced by quantity discounts.

4.4 Inventory Models

1. **Deterministic Models**:
    - Assume demand and lead times are known and constant.
    - Common deterministic models:
        - **Economic Order Quantity (EOQ)**: Determines the optimal order size to minimize total costs. EOQ=2DSHEOQ = \sqrt{\frac{2DS}{H}} where:
            - DD = Annual demand
            - SS = Ordering cost per order
            - HH = Holding cost per unit per year
        - **Economic Production Quantity (EPQ)**: Extends EOQ to include production scenarios.
2. **Stochastic Models**:
    - Incorporate uncertainty in demand or lead time.
    - Focus on determining reorder points and safety stock levels.
    - Examples:

        - **(Q, R) Policy**: Fixed order quantity (QQ) placed when inventory falls below a reorder point (RR).
        - **(s, S) Policy**: Inventory is replenished to level SS when it drops below ss.
3. **Multi-Period Models**:
    - Address long-term inventory management over multiple time periods.
    - Example: Dynamic Programming approaches like the Wagner-Whitin model.
4. **Single-Period Models**:
    - Focus on one-time ordering decisions, such as in seasonal goods.
    - Example: Newsvendor model to balance overstock and understock costs.

4.5 Applications of Inventory Theory

- **Retail**: Managing stock levels to meet customer demand while minimizing overstock.
- **Manufacturing**: Ensuring raw material availability for uninterrupted production.
- **Supply Chain Management**: Coordinating inventory across multiple locations.
- **Healthcare**: Managing medical supplies and pharmaceuticals.
- **E-commerce**: Balancing inventory for rapid order fulfillment and cost-efficiency.

Benefits of Inventory Theory

- Reduces operational costs.
- Improves customer satisfaction by avoiding stockouts.
- Enhances decision-making through systematic planning.
- Provides flexibility to respond to market changes or supply disruptions.

By employing inventory theory, organizations can achieve an optimal balance between inventory availability and cost-efficiency, ensuring both profitability and customer satisfaction.

Demand Rate in Inventory Context

- The rate of service (μ\mu) calculated above is equal to the demand rate (λ\lambda) for the inventory model.
- In the context of inventory management, the demand rate (r) corresponds to the service rate: r=λ=μr = \lambda = \mu

4.6 Inter-Inventory Loading Rate and Inter-Arrival Rate

Objective:

- Extend the approach to calculate inter-inventory loading rate (k) and inter-arrival rate (λ\lambda) in a queuing model.

Key Insight:
The inter-arrival rate and inter-inventory loading rate can be treated similarly:

k=λk

This implies the demand or arrival rate governs both inventory restocking and customer service operations.

4.7 Exploring Inventory Parameters

Inventory Parameters

- r: Demand rate.
- k: Loading rate.
- t: Reorder time interval.
- n: Number of inventory cycles.
- q: Order quantity.
- D: Demand during the lead time.
- C1: Cost per unit of inventory.
- C0: Ordering/setup cost.

4.8 Estimating Queue Characteristics

Key Metrics:

a. **Probability of no transactions (P0):**

$P0=1-\lambda\mu$

b. **Probability of at least one transaction (P(≥1)):**

$P(\geq 1)=1-P0$

c. **Expected idle time of the shop (idleT):**
Based on the probability of no transactions:

$Tidle=P0 \cdot Total\ TimeT$

d. **Expected number of sales (L):**

$L=\lambda\mu-\lambda L$

e. **Expected waiting time (W):**

$W=1\mu-\lambda W$

a. Optimal Order Quantity (q0):

Using the Economic Order Quantity (EOQ) formula:

$q0=2 \cdot D \cdot C0C1$

b. Optimal Number of Inventory Runs (n0):
$n0=Dq0$

c. Optimal Replacing Time (t0):
t0=q0r

d. Optimal Total Expected Inventory Cost:
Total Cost=D·C12·q0+n0·C0\text{Total Cost}

R Codes and Output for the Inventory Analysis

```
> # Inventory parameters

> inventory_params <- list(
+   r = demand_rate,  # Demand rate
+   k = inventory_loading_rate,  # Loading rate
+   t = avg_inter_time,  # Average inter-arrival time
+   n = length(unique(data$Date)),  # Number of days in the dataset
+   q = 50,  # Example reorder quantity (to be adjusted based on
context)
+   D = sum(data$Quantity, na.rm = TRUE),  # Total demand
+   C1 = 2,  # Example holding cost per unit
+   C0 = 100  # Example ordering cost
+ )
> # Display queueing and inventory parameters
> cat("Queueing Parameters:\n")
Queueing Parameters:
> print(queue_params)
$lambda
[1] 0.0003374617

$mu
[1] 0.000404954

$C
[1] 1

$N
[1] 178
```

Task : Inventory management outcomes

```
> optimal_order_quantity <- sqrt(2 * inventory_params$D *
inventory_params$C0 / inventory_params$C1)  # EOQ formula
> optimal_inventory_runs <- inventory_params$D /
optimal_order_quantity
> optimal_cycle_time <- 1 / inventory_params$r  # Average cycle
length
> optimal_inventory_cost <- (optimal_order_quantity / 2) *
inventory_params$C1 + inventory_params$C0 * optimal_inventory_runs
> # Display inventory management outcomes
> cat("Optimal Order Quantity (q0):", optimal_order_quantity, "\n")
Optimal Order Quantity (q0): 300.3331
> cat("Optimal Number of Inventory Runs (n0):",
optimal_inventory_runs, "\n")
Optimal Number of Inventory Runs (n0): 3.003331
> cat("Optimal Cycle Time (t0):", optimal_cycle_time, "\n")
Optimal Cycle Time (t0): 2963.299
> cat("Optimal Total Inventory Cost:", optimal_inventory_cost, "\n")
Optimal Total Inventory Cost: 600.6663
```

Chapter 5: Summary and Report:

Summary Report on Queueing and Inventory Analysis on Fashion Accessories

This report summarizes the key findings and calculations for queueing and inventory management tasks based on the provided data. The steps include evaluating inter-transaction times, estimating service rates, selecting an appropriate queuing model, and calculating optimal inventory parameters.

1. Chapter 3- Queueing Analysis

Average Inter-Transaction Time (Task 11)

- **Average Inter-Transaction Time (minutes):** 2963.299
- **Service Rate (λ):** 0.0003374617 (calculated as 1/Mean Time)

Selected Queueing Model (Task 15)

- Based on the parameters:
  - Single server (C = 1)
  - Finite transactions (N=178)
- **Model:** M/M/1:(N/FIFO)M/M/1: (N/FIFO)

Queue Characteristics (Task 16)

- **Probability of no sales (P(0)):** 0.43459820.4345982
- **Probability of at least one sale (P(at least 1)):** 0.56540180.5654018
- **Expected idle time:** 2469.4162469.416 minutes
- **Expected number of completed sales:** 11 sales
- **Expected waiting time:** 14816.514816.5 minutes

2. Chapter 4- Inventory Management

Demand and Inventory Loading Rates

- **Demand Rate (r = λ):** 0.0003374617
- **Inventory Loading Rate (k=λk = λ):** 0.00033746170

Inventory Parameters

Key parameters for inventory calculations include:

- **Reorder Quantity (q):** 5050 (example value)
- **Holding Cost (C1):** 22
- **Ordering Cost (C0):** 100100
- **Total Demand (D):** 178178 units
- **Number of Days in Dataset (n):** Calculated as unique transaction days.

Optimal Inventory Outcomes

- **Optimal Order Quantity (q0):** 300.3331 units
  (Derived using the EOQ formula: (2·D·C0)/C1\sqrt{(2 \cdot D \cdot C0)/C1})
- **Optimal Number of Inventory Runs (n0):** 3.0033313.003331
  (Derived as D/q0)
- **Optimal Cycle Time (t0):** 2963.299 minutes
  (Derived as 1/r)
- **Optimal Total Inventory Cost:** 600.6663

Key Insights and Conclusions

1. **Queueing System:** The selected queueing model, M/M/1:(N/FIFO)M/M/1: (N/FIFO), represents a single-server system with limited transaction capacity. Key probabilities, idle times, and expected waiting times align with the calculated service rate. The system handles transactions with a probability of at least one sale being 0.5654.
2. **Inventory Management:** The Economic Order Quantity (EOQ) framework ensures that the shop minimizes total inventory costs while maintaining a balance between ordering and holding costs. The optimal values for inventory management suggest a feasible strategy to reduce costs and meet demand.

By integrating queueing and inventory models, the analysis demonstrates an efficient system for managing both transactions and inventory in a business context.

Theoretical Report:
Sales and Inventory Data Analysis for Fashion Accessories

1. Objective

The goal of this analysis is to provide actionable insights into sales performance, customer behavior, and inventory management using transactional data. The report also integrates queueing theory to understand service dynamics.

2. Key Findings

Sales Trends Over Time
- Insight: Sales showed significant fluctuations across the time period, with peak revenues occurring on specific dates.
- Actionable Recommendation: Identify and replicate successful marketing strategies from peak periods for other dates.

Revenue by Product Line
- Insigh*: Product lines such as "Fashion Accessories" and "Food and Beverages" contributed the highest revenues.
- Actionable Recommendation: Allocate more shelf space and promotions to high-revenue product lines.

Revenue by Branch
-Insight: Revenue generation varied significantly across branches. Some branches consistently outperformed others.
- Actionable Recommendation: Investigate the operational practices of top-performing branches and replicate them in underperforming ones.

Payment Method Distribution
- Insight: The distribution of payment methods (Cash, Credit Card, E-wallet) was nearly balanced, with a slight preference for cash payments.
- Actionable Recommendation: Offer incentives like cashback for digital payments to further boost non-cash transactions.

Customer Ratings Distribution
-Insight: Most customer ratings were concentrated around the average score, with occasional extreme ratings.
- Actionable Recommendation: Collect qualitative feedback from customers to address areas causing dissatisfaction and reinforce positive aspects.

3.Queueing and Inventory Management Analysis

Queueing Model Analysis
- Model Used: M/M/1: (N/FIFO), representing a single-server system with finite capacity.
- Key Metrics:
  - Probability of Idle System: 43.46%.
  - Expected Waiting Time: Approximately 14,816 minutes.
  - Expected Completed Transactions: 1 transaction within the observation period.
- Actionable Recommendation: Increase staffing or service efficiency to reduce customer waiting times during peak hours.

Inventory Optimization
- Demand Rate: 0.00034 units/minute (based on transaction data).
- Optimal Order Quantity (EOQ): 300 units per order.
- Optimal Inventory Cost: $600.67 per cycle.
-Actionable Recommendation: Implement EOQ principles to minimize holding and ordering costs while maintaining sufficient stock levels.

4. Action Plan

1. Sales Strategies:
- Replicate successful promotional tactics from high-sales dates.
- Focus advertising campaigns on high-revenue product lines.

2. Operational Improvements:
- Analyze customer flow and staff scheduling to address peak wait times.
- Train staff in underperforming branches to mirror practices from successful ones.

3. Payment Method Innovations:
- Encourage digital transactions by offering loyalty points or discounts.

4. Customer Experience:
- Implement a feedback system to capture real-time customer sentiments.

5. Inventory and Logistics:
- Use EOQ models for restocking decisions.
- Monitor demand trends and adjust replenishment schedules dynamically.

5. Conclusion

This analysis highlights opportunities to improve both sales performance and operational efficiency. By aligning inventory strategies with demand patterns and optimizing service operations, the business can achieve sustainable growth and enhanced customer satisfaction.

Bibliography:

Here are some books that cover the basics of queueing and inventory modeling, suitable for students and practitioners alike:

Queueing Theory

1. "Fundamentals of Queueing Theory" by Donald Gross, John F. Shortle, James M. Thompson, and Carl M. Harris
2. "Stochastic Processes" by J. Medhi
3. "Queueing Systems, Volume 1: Theory" by Leonard Kleinrock

Inventory Modeling

1. "Principles of Inventory Management: When You Are Down to Four, Order More" by John A. Muckstadt and Amar Sapra
2. "Inventory Control and Management" by Donald Waters
3. "Analysis of Inventory Systems" by George Hadley and Thomas M. Whitin

Appendix:

```
# Load necessary library
library(readxl)  # To read Excel file

# Step 1: Load the data
library(readxl)
dataset_queueing <- read_excel("dataset queueing.xlsx")
View(dataset_queueing)

# Step 2: Preprocess the data
# Extract necessary columns and group by Invoice ID
transactions_data <- aggregate(`Product line` ~ `Invoice ID`,
dataset_queueing, paste, collapse = ",")
transactions_data

# Step 3: Convert to transaction format
>transactions_list <- strsplit(transactions_data$`Product line`, ",")
>transactions_list
> unique_items <- unique(unlist(transactions_list))
> unique_items
# Create a binary transaction matrix
> transaction_matrix <- sapply(unique_items, function(item) {
 sapply(transactions_list, function(transaction) item %in%
transaction)
})
> transaction_matrix <- as.data.frame(transaction_matrix)
> transaction_matrix
# Step 4: Generate summary
# Calculate item frequency
> item_frequency <- colSums(transaction_matrix)
> item_frequency <- sort(item_frequency, decreasing = TRUE)
# 1. Sales trends over time
> windows(width = 10, height = 7)
> sales_trend <- aggregate(Total ~ Date, data = data, sum)
> plot(sales_trend$Date, sales_trend$Total, type = "o", col = "blue",
+      main = "Sales Trends Over Time", xlab = "Date", ylab = "Total
Revenue ($)")
# Total revenue by branch
> par(mar = c(5, 5, 4, 2))
> revenue_by_branch <- aggregate(Total ~ Branch, data = data, sum)
> barplot(revenue_by_branch$Total, names.arg =
revenue_by_branch$Branch,
+         col = "purple", main = "Revenue by Branch", ylab = "Total
Revenue ($)")
Distribution of payment methods
> par(mar = c(4, 4, 4, 4))
> payment_distribution <- table(data$Payment)
> pie(payment_distribution, col = c("skyblue", "pink", "lightgreen"),
    +     main = "Payment Method Distribution")
```

```
#Customer ratings distribution
> par(mar = c(5, 5, 4, 2))
> hist(data$Rating, breaks = 10, col = "orange",
+      main = "Customer Ratings Distribution", xlab = "Rating", ylab
= "Frequency")

# Print top 3 items
> # Print top 3 items
> cat("Top 3 items:\n")
> print(head(item_frequency, 3))
> # Step 5: Display association (manual inspection)
> # Example: Checking co-occurrence of top 2 items
> top_items <- names(head(item_frequency, 2))
> co_occurrence <- rowSums(transaction_matrix[, top_items])
> cat("Co-occurrence of", top_items[1], "and", top_items[2], ":\n")
> print(sum(co_occurrence == 2))
> # Step 6: Save results
> write.csv(item_frequency, "item_frequency.csv", row.names = FALSE)
> # Load the data
> library(readxl)
> data <- read_excel("fashion.xlsx")
> View(data)
> # Ensure Date is in the correct format
> data$Date <- as.Date(data$Date, format = "%Y-%m-%d")
> # Ensure Time is in HH:MM:SS format; check if Time needs formatting
> data$Time <- format(hms::as_hms(data$Time), "%H:%M:%S")
> # Combine Date and Time into a single POSIXct DateTime object
> data$DateTime <- as.POSIXct(paste(data$Date, data$Time), format =
"%Y-%m-%d %H:%M:%S")
> data$DateTime
# Extract Year, Month (numeric and name), Week, and Day
> data$Year <- format(data$DateTime, "%Y")
> data$Month <- format(data$DateTime, "%m")  # Numeric month
> data$Month_Name <- format(data$DateTime, "%B")  # Full month name
> data$Week <- format(data$DateTime, "%U")  # Week number
> data$Day <- format(data$DateTime, "%d")  # Day of the month
> data$Day_Name <- format(data$DateTime, "%A")  # Day name
> # Task 2: Extract Hours, Minutes, Seconds and calculate total
seconds
> data$Hour <- format(data$DateTime, "%H")
> data$Minute <- format(data$DateTime, "%M")
> data$Second <- format(data$DateTime, "%S")
> # Calculate total seconds since the start of the day
> data$Total_Seconds <- as.numeric(data$Hour) * 3600 +
as.numeric(data$Minute) * 60 + as.numeric(data$Second)
> data$Total_Seconds
# Convert total seconds to minutes
> data$Total_Minutes <- data$Total_Seconds / 60
> data$Total_Minutes
# Task 3: Find the inter-transaction time between
consecutive transactions
> # Sort data by DateTime
> data <- data[order(data$DateTime), ]
> data
```

```
# Calculate inter-transaction time in seconds
> data$Inter_Transaction_Time <- c(NA,
diff(as.numeric(data$DateTime)))
> data$Inter_Transaction_Time
# Task: Calculate the average inter-transaction time
for the session or day
> avg_inter_time <- mean(data$Inter_Transaction_Time, na.rm = TRUE)
> avg_inter_time
> service_rate <- 1 / avg_inter_time  # Service rate (lambda)
> service_rate
> avg_inter_time <- mean(data$Inter_Transaction_Time/60, na.rm =
TRUE)
> avg_inter_time
> service_rate <- 1 / avg_inter_time  # Service rate (lambda)
> service_rate
# Display results
> cat("Average Inter-Transaction Time (minutes):", avg_inter_time,
"\n")
> cat("Service Rate (1/Mean Time):", service_rate, "\n")
# Task: Set service rate as demand rate (r = lambda)
> demand_rate <- service_rate
> cat("Demand Rate (r):", demand_rate, "\n")
> # Task 12: Set service rate as demand rate (r = lambda)
> demand_rate <- service_rate
> cat("Demand Rate (r):", demand_rate, "\n")
> # Task : Calculate inventory loading rate (k =
lambda)
> inventory_loading_rate <- service_rate  # k = lambda
> cat("Inventory Loading Rate (k):", inventory_loading_rate, "\n")

# Task : Explore queueing and inventory parameters
> # Queueing parameters
> queue_params <- list(
+   lambda = service_rate,  # Arrival rate
+   mu = service_rate * 1.2,  # Assuming service rate is 20% faster
than arrival rate
+   C = 1,  # Single server
+   N = nrow(data)  # Total number of transactions
+ )
> # Inventory parameters
> inventory_params <- list(
+   r = demand_rate,  # Demand rate
+   k = inventory_loading_rate,  # Loading rate
+   t = avg_inter_time,  # Average inter-arrival time
+   n = length(unique(data$Date)),  # Number of days in the dataset
+   q = 50,  # Example reorder quantity (to be adjusted based on
context)
+   D = sum(data$Quantity, na.rm = TRUE),  # Total demand
+   C1 = 2,  # Example holding cost per unit
+   C0 = 100  # Example ordering cost
+ )
> # Display queueing and inventory parameters
```

```
> cat("Queueing Parameters:\n")
> print(queue_params)
# Task : Identify the appropriate queueing model
> if (queue_params$C == 1 && is.infinite(queue_params$N)) {
+   queue_model <- "M/M/1: (∞/FIFO)"
+ } else if (queue_params$C == 1 && !is.infinite(queue_params$N)) {
+   queue_model <- "M/M/1: (N/FIFO)"
+ } else if (queue_params$C > 1 && is.infinite(queue_params$N)) {
+   queue_model <- "M/M/C: (∞/FIFO)"
+ } else {
+   queue_model <- "M/M/C: (N/FIFO)"
+ }
> cat("Selected Queueing Model:", queue_model, "\n")
# Task : Estimate queue characteristics
> prob_no_sales <- exp(-queue_params$lambda / queue_params$mu)  #
P(0)
> prob_at_least_one_sale <- 1 - prob_no_sales  # P(at least 1)
> idle_time <- 1 / queue_params$mu  # Expected idle time
> completed_sales <- queue_params$lambda * avg_inter_time  # Expected
number of sales
> waiting_time <- 1 / (queue_params$mu - queue_params$lambda)  #
Expected waiting time
> # Display queue characteristics
> cat("Probability of no sales:", prob_no_sales, "\n")
> cat("Probability of at least one sale:", prob_at_least_one_sale,
"\n")
> cat("Expected idle time:", idle_time, "\n")
> cat("Expected number of completed sales:", completed_sales, "\n")
> cat("Expected waiting time:", waiting_time, "\n")
# Task : Inventory management outcomes
> optimal_order_quantity <- sqrt(2 * inventory_params$D *
inventory_params$C0 / inventory_params$C1)  # EOQ formula
> optimal_inventory_runs <- inventory_params$D /
optimal_order_quantity
> optimal_cycle_time <- 1 / inventory_params$r  # Average cycle
length
> optimal_inventory_cost <- (optimal_order_quantity / 2) *
inventory_params$C1 + inventory_params$C0 * optimal_inventory_runs
> # Display inventory management outcomes
> cat("Optimal Order Quantity (q0):", optimal_order_quantity, "\n")
> cat("Optimal Number of Inventory Runs (n0):",
optimal_inventory_runs, "\n")
> cat("Optimal Cycle Time (t0):", optimal_cycle_time, "\n")
> cat("Optimal Total Inventory Cost:", optimal_inventory_cost, "\n")
```

Dataset:

Invoice ID	Branch	City	Customer	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time
765-26-6951	A	Yangon	Normal	Sports and travel	72.61	6	21.783	457.443	01-01-2023	10:39
530-90-9855	A	Yangon	Member	Home and lifestyle	47.59	8	19.036	399.756	01-01-2023	14:47
891-01-7034	B	Mandalay	Normal	Electronic accessories	74.71	6	22.413	470.673	01-01-2023	19:07
493-65-6248	C	Naypyitaw	Member	Sports and travel	36.98	10	18.49	388.29	01-01-2023	19:48
556-97-7101	C	Naypyitaw	Normal	Electronic accessories	63.22	2	6.322	132.762	01-01-2023	15:51
133-14-7229	C	Naypyitaw	Normal	Health and beauty	62.87	2	6.287	132.027	01-01-2023	11:43
651-88-7328	A	Yangon	Normal	Fashion accessories	65.74	9	29.583	621.243	01-01-2023	13:55
182-52-7000	A	Yangon	Member	Sports and travel	27.04	4	5.408	113.568	01-01-2023	20:26
416-17-9926	A	Yangon	Member	Electronic accessories	74.22	10	37.11	779.31	01-01-2023	14:42
271-77-8740	C	Naypyitaw	Member	Sports and travel	29.22	6	8.766	184.086	01-01-2023	11:40
770-42-8960	B	Mandalay	Normal	Food and beverages	21.12	8	8.448	177.408	01-01-2023	19:31
746-04-1077	B	Mandalay	Member	Food and beverages	84.63	10	42.315	888.615	01-01-2023	11:36
551-21-3069	C	Naypyitaw	Normal	Electronic accessories	23.07	9	10.3815	218.0115	02-01-2023	11:27
237-01-6122	C	Naypyitaw	Member	Home and lifestyle	80.79	9	36.3555	763.4655	02-01-2023	20:31
105-31-1824	A	Yangon	Member	Sports and travel	69.52	7	24.332	510.972	02-01-2023	15:10
870-76-1733	A	Yangon	Member	Food and beverages	14.23	5	3.5575	74.7075	02-01-2023	10:08
358-88-9262	C	Naypyitaw	Member	Food and beverages	87.48	6	26.244	551.124	02-01-2023	18:43
443-82-0585	A	Yangon	Member	Health and beauty	77.68	4	15.536	326.256	02-01-2023	19:54
189-98-2939	C	Naypyitaw	Normal	Fashion accessories	78.55	9	35.3475	742.2975	03-01-2023	13:22
453-63-6187	B	Mandalay	Normal	Electronic accessories	27.5	3	4.125	86.625	03-01-2023	15:40
153-58-4872	C	Naypyitaw	Member	Food and beverages	74.89	4	14.978	314.538	03-01-2023	15:32
220-28-1851	A	Yangon	Normal	Home and lifestyle	34.73	2	3.473	72.933	03-01-2023	18:14
339-96-8318	B	Mandalay	Member	Fashion accessories	81.31	7	28.4585	597.6285	03-01-2023	19:49
207-73-1363	B	Mandalay	Normal	Health and beauty	69.51	2	6.951	145.971	03-01-2023	12:15
470-31-3286	B	Mandalay	Normal	Health and beauty	14.82	3	2.223	46.683	03-01-2023	11:30
811-03-8790	A	Yangon	Normal	Electronic accessories	45.48	10	22.74	477.54	03-01-2023	10:22
247-11-2470	A	Yangon	Member	Fashion accessories	22.32	4	4.464	93.744	03-01-2023	16:23
238-45-6950	B	Mandalay	Member	Food and beverages	53.72	1	2.686	56.406	03-01-2023	20:03
504-35-8843	A	Yangon	Normal	Sports and travel	42.47	1	2.1235	44.5935	01-02-2023	16:57
446-47-6729	C	Naypyitaw	Normal	Fashion accessories	99.82	2	9.982	209.622	01-02-2023	18:09
244-08-0162	B	Mandalay	Normal	Health and beauty	34.21	10	17.105	359.205	01-02-2023	13:00
198-84-7132	B	Mandalay	Member	Fashion accessories	40.61	9	18.2745	383.7645	01-02-2023	13:40

720-72-2436	A	Yangon	Normal	Food and beverages	66.52	4	13.304	279.384	03-02-2023	18:14
172-42-8274	B	Mandalay	Normal	Electronic accessories	38.27	2	3.827	80.367	03-02-2023	18:18
303-96-2227	B	Mandalay	Normal	Home and lifestyle	97.38	10	48.69	1022.49	03-02-2023	17:16
249-42-3782	A	Yangon	Normal	Health and beauty	70.01	5	17.5025	367.5525	01-03-2023	11:36
749-24-1565	A	Yangon	Normal	Health and beauty	23.03	9	10.3635	217.6335	01-03-2023	12:02
687-15-1097	C	Naypyitaw	Member	Health and beauty	21.12	2	2.112	44.352	01-03-2023	19:17
422-29-8786	A	Yangon	Normal	Home and lifestyle	67.09	5	16.7725	352.2225	01-03-2023	16:47
343-87-0864	C	Naypyitaw	Member	Health and beauty	75.88	1	3.794	79.674	01-03-2023	10:30
875-31-8302	B	Mandalay	Normal	Sports and travel	93.38	1	4.669	98.049	01-03-2023	13:07
501-61-1753	B	Mandalay	Normal	Home and lifestyle	63.15	6	18.945	397.845	01-03-2023	20:24
552-44-5977	B	Mandalay	Member	Health and beauty	62	8	24.8	520.8	01-03-2023	19:08
326-78-5178	C	Naypyitaw	Member	Food and beverages	91.4	7	31.99	671.79	02-03-2023	10:19
347-34-2234	B	Mandalay	Member	Sports and travel	55.07	9	24.7815	520.4115	02-03-2023	13:40
430-60-3493	A	Yangon	Member	Home and lifestyle	94.88	7	33.208	697.368	02-03-2023	14:38
566-71-1091	A	Yangon	Normal	Fashion accessories	77.02	5	19.255	404.355	02-03-2023	15:59
639-76-1242	C	Naypyitaw	Normal	Food and beverages	40.52	5	10.13	212.73	02-03-2023	15:19
326-71-2155	C	Naypyitaw	Normal	Sports and travel	73.95	4	14.79	310.59	02-03-2023	10:02
277-63-2961	B	Mandalay	Member	Sports and travel	73.97	1	3.6985	77.6685	02-03-2023	15:53
718-57-9773	C	Naypyitaw	Normal	Sports and travel	49.33	10	24.665	517.965	02-03-2023	16:40
443-59-0061	A	Yangon	Member	Food and beverages	67.45	10	33.725	708.225	02-03-2023	11:25
742-04-5161	A	Yangon	Member	Home and lifestyle	72.78	10	36.39	764.19	02-03-2023	17:24
322-02-2271	B	Mandalay	Normal	Sports and travel	42.97	3	6.4455	135.3555	02-03-2023	11:46
585-86-8361	A	Yangon	Normal	Food and beverages	27.28	5	6.82	143.22	02-03-2023	10:31
190-59-3964	B	Mandalay	Member	Food and beverages	47.16	5	11.79	247.59	02-03-2023	14:35
676-10-2200	B	Mandalay	Member	Fashion accessories	53.78	1	2.689	56.469	02-03-2023	20:13
631-41-3108	A	Yangon	Normal	Home and lifestyle	46.33	7	16.2155	340.5255	03-03-2023	13:23
777-82-7220	B	Mandalay	Member	Home and lifestyle	30.12	8	12.048	253.008	03-03-2023	13:01
861-77-0145	C	Naypyitaw	Member	Electronic accessories	81.97	10	40.985	860.685	03-03-2023	14:30
733-33-4967	C	Naypyitaw	Normal	Electronic accessories	20.85	8	8.34	175.14	03-03-2023	19:17
175-54-2529	A	Yangon	Member	Food and beverages	22.17	8	8.868	186.228	03-03-2023	17:01
305-14-0245	B	Mandalay	Member	Home and lifestyle	94.49	8	37.796	793.716	03-03-2023	19:00
211-05-0490	C	Naypyitaw	Member	Electronic accessories	51.92	5	12.98	272.58	03-03-2023	13:42
568-88-3448	A	Yangon	Normal	Health and beauty	25	1	1.25	26.25	03-03-2023	15:09

720-72-2436	A	Yangon	Normal	Food and beverages	66.52	4	13.304	279.384	03-02-2023	18:14
172-42-8274	B	Mandalay	Normal	Electronic accessories	38.27	2	3.827	80.367	03-02-2023	18:18
303-96-2227	B	Mandalay	Normal	Home and lifestyle	97.38	10	48.69	1022.49	03-02-2023	17:16
249-42-3782	A	Yangon	Normal	Health and beauty	70.01	5	17.5025	367.5525	01-03-2023	11:36
749-24-1565	A	Yangon	Normal	Health and beauty	23.03	9	10.3635	217.6335	01-03-2023	12:02
687-15-1097	C	Naypyitaw	Member	Health and beauty	21.12	2	2.112	44.352	01-03-2023	19:17
422-29-8786	A	Yangon	Normal	Home and lifestyle	67.09	5	16.7725	352.2225	01-03-2023	16:47
343-87-0864	C	Naypyitaw	Member	Health and beauty	75.88	1	3.794	79.674	01-03-2023	10:30
875-31-8302	B	Mandalay	Normal	Sports and travel	93.38	1	4.669	98.049	01-03-2023	13:07
501-61-1753	B	Mandalay	Normal	Home and lifestyle	63.15	6	18.945	397.845	01-03-2023	20:24
552-44-5977	B	Mandalay	Member	Health and beauty	62	8	24.8	520.8	01-03-2023	19:08
326-78-5178	C	Naypyitaw	Member	Food and beverages	91.4	7	31.99	671.79	02-03-2023	10:19
347-34-2234	B	Mandalay	Member	Sports and travel	55.07	9	24.7815	520.4115	02-03-2023	13:40
430-60-3493	A	Yangon	Member	Home and lifestyle	94.88	7	33.208	697.368	02-03-2023	14:38
566-71-1091	A	Yangon	Normal	Fashion accessories	77.02	5	19.255	404.355	02-03-2023	15:59
639-76-1242	C	Naypyitaw	Normal	Food and beverages	40.52	5	10.13	212.73	02-03-2023	15:19
326-71-2155	C	Naypyitaw	Normal	Sports and travel	73.95	4	14.79	310.59	02-03-2023	10:02
277-63-2961	B	Mandalay	Member	Sports and travel	73.97	1	3.6985	77.6685	02-03-2023	15:53
718-57-9773	C	Naypyitaw	Normal	Sports and travel	49.33	10	24.665	517.965	02-03-2023	16:40
443-59-0061	A	Yangon	Member	Food and beverages	67.45	10	33.725	708.225	02-03-2023	11:25
742-04-5161	A	Yangon	Member	Home and lifestyle	72.78	10	36.39	764.19	02-03-2023	17:24
322-02-2271	B	Mandalay	Normal	Sports and travel	42.97	3	6.4455	135.3555	02-03-2023	11:46
585-86-8361	A	Yangon	Normal	Food and beverages	27.28	5	6.82	143.22	02-03-2023	10:31
190-59-3964	B	Mandalay	Member	Food and beverages	47.16	5	11.79	247.59	02-03-2023	14:35
676-10-2200	B	Mandalay	Member	Fashion accessories	53.78	1	2.689	56.469	02-03-2023	20:13
631-41-3108	A	Yangon	Normal	Home and lifestyle	46.33	7	16.2155	340.5255	03-03-2023	13:23
777-82-7220	B	Mandalay	Member	Home and lifestyle	30.12	8	12.048	253.008	03-03-2023	13:01
861-77-0145	C	Naypyitaw	Member	Electronic accessories	81.97	10	40.985	860.685	03-03-2023	14:30
733-33-4967	C	Naypyitaw	Normal	Electronic accessories	20.85	8	8.34	175.14	03-03-2023	19:17
175-54-2529	A	Yangon	Member	Food and beverages	22.17	8	8.868	186.228	03-03-2023	17:01
305-14-0245	B	Mandalay	Member	Home and lifestyle	94.49	8	37.796	793.716	03-03-2023	19:00
211-05-0490	C	Naypyitaw	Member	Electronic accessories	51.92	5	12.98	272.58	03-03-2023	13:42
568-88-3448	A	Yangon	Normal	Health and beauty	25	1	1.25	26.25	03-03-2023	15:09