COMP1013 Project T3 2025

Thien Nhan Pham - 22097306

2026-01-21

#Declaration

By including this statement, we the authors of this work, verify that:

- We hold a copy of this assignment that we can produce if the original is lost or damaged.

- We hereby certify that no part of this assignment/product has been copied from any other student's work or from any other source except where due acknowledgement is made in the assignment.

- No part of this assignment/product has been written/produced for us by another person except where such collaboration has been authorised by the subject lecturer/tutor concerned.

- We are aware that this work may be reproduced and submitted to plagiarism detection software programs for the purpose of detecting possible plagiarism (which may retain a copy on its database for future plagiarism checking).

- We hereby certify that we have read and understand what the School of Computing, Engineering and Mathematics defines as minor and substantial breaches of misconduct as outlined in the learning guide for this unit.

```r
# sales_ug.csv: main dataset (store, product, day level)
# product_hierarchy.csv: product_id -> hierarchy categories
# store_cities.csv: store metadata (type, size, city)
sales <- read.csv("sales_ug.csv")
product_hierarchy <- read.csv("product_hierarchy.csv")
store_cities <- read.csv("store_cities.csv")

# Convert date column into Date type (important for correct plotting and grouping)
sales$date <- as.Date(sales$date)

# str() shows column names + data types + a preview of values
str(sales)
```

```
## 'data.frame':    104000 obs. of  11 variables:
##  $ product_id          : chr  "P0001" "P0001" "P0001" "P0001" ...
##  $ store_id            : chr  "S0002" "S0038" "S0040" "S0050" ...
##  $ date                : Date, format: "2017-07-03" "2017-07-03" ...
##  $ sales               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ revenue             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ stock               : num  1 1 2 1 10 5 24 25 7 3 ...
##  $ price               : num  6.75 6.75 6.75 6.75 6.75 6.75 349 349 4.5 33.9 ...
##  $ promo_type_1        : chr  "PR14" "PR14" "PR14" "PR14" ...
##  $ promo_bin_1         : chr  "" "" "" "" ...
##  $ promo_discount_2    : logi  NA NA NA NA NA NA ...
##  $ promo_discount_type_2: logi  NA NA NA NA NA NA ...
```

```r
str(product_hierarchy)
```

```
## 'data.frame':    699 obs. of  10 variables:
##  $ product_id    : chr  "P0000" "P0001" "P0002" "P0004" ...
##  $ product_length: num  5 13.5 22 2 16 8.5 2 5 5 2 ...
##  $ product_depth : num  20 22 40 13 30 15 22 16 18 22 ...
##  $ product_width : num  12 20 22 4 16 15 9.5 5 14 3 ...
##  $ cluster_id    : chr  "" "cluster_5" "cluster_0" "cluster_3" ...
##  $ hierarchy1_id : chr  "H00" "H01" "H03" "H03" ...
##  $ hierarchy2_id : chr  "H0004" "H0105" "H0315" "H0314" ...
##  $ hierarchy3_id : chr  "H000401" "H010501" "H031508" "H031405" ...
##  $ hierarchy4_id : chr  "H00040105" "H01050100" "H03150800" "H03140500" ...
##  $ hierarchy5_id : chr  "H0004010534" "H0105010006" "H0315080028" "H0314050003" ...
```

```r
str(store_cities)
```

```
## 'data.frame':    144 obs. of  4 variables:
##  $ store_id    : chr  "S0091" "S0012" "S0045" "S0032" ...
##  $ storetype_id: chr  "ST04" "ST04" "ST04" "ST03" ...
##  $ store_size  : int  19 28 17 14 24 20 44 24 14 19 ...
##  $ city_id     : chr  "C013" "C005" "C008" "C019" ...
```

```r
#   - Missing values can break calculations or bias averages
#   - We need to know if key columns have NA before analysis
# colSums(is.na()) counts how many NA values are in each column.nice_table(as.data.frame(colSums(is.na(
nice_table(data.frame(missing = colSums(is.na(product_hierarchy))),"Missing values in product hierarchy
```

Table 1: Missing values in product hierarchy

	missing
product_id	0
product_length	18
product_depth	16
product_width	16
cluster_id	0
hierarchy1_id	0
hierarchy2_id	0
hierarchy3_id	0
hierarchy4_id	0
hierarchy5_id	0

```r
nice_table(as.data.frame(colSums(is.na(store_cities))), "Missing values in store cities")
```

Table 2: Missing values in store cities

	colSums(is.na(store_cities))
store_id	0
storetype_id	0

store_size	0
city_id	0

==

TASK 1: Store revenue across days

==

The sales dataset records revenue at the (store, product, day) level. To compute total revenue per store at the end of each day, revenue is summed across all products for each store and date. To compare differences between days, daily totals are summed across stores. Weekly totals are obtained by summing store daily revenue across the full seven-day period, then visualised.

```r
# group_by(store_id, date) -> group rows by store and day
# summarise(sum(revenue))  -> total revenue per store per day
# sales is at (store, product, date) level.
# That means each row is ONE product sold at ONE store on ONE day.
# To get daily store revenue, we must sum revenue across all products per store per day.
store_daily_revenue <- sales %>%
group_by(store_id, date) %>%
summarise(total_revenue = sum(revenue, na.rm = TRUE), .groups = "drop") %>%
arrange(store_id, date)
# Show first rows for checking
nice_table(head(store_daily_revenue, 10), "First 10 rows: daily total revenue per store")
```

Table 3: First 10 rows: daily total revenue per store

store_id	date	total_revenue
S0001	2017-07-03	767.99
S0001	2017-07-04	1296.36
S0001	2017-07-05	1005.85
S0001	2017-07-06	893.55
S0001	2017-07-07	1247.88
S0001	2017-07-08	1547.33
S0001	2017-07-09	1465.23
S0002	2017-07-03	346.82
S0002	2017-07-04	226.18
S0002	2017-07-05	175.48

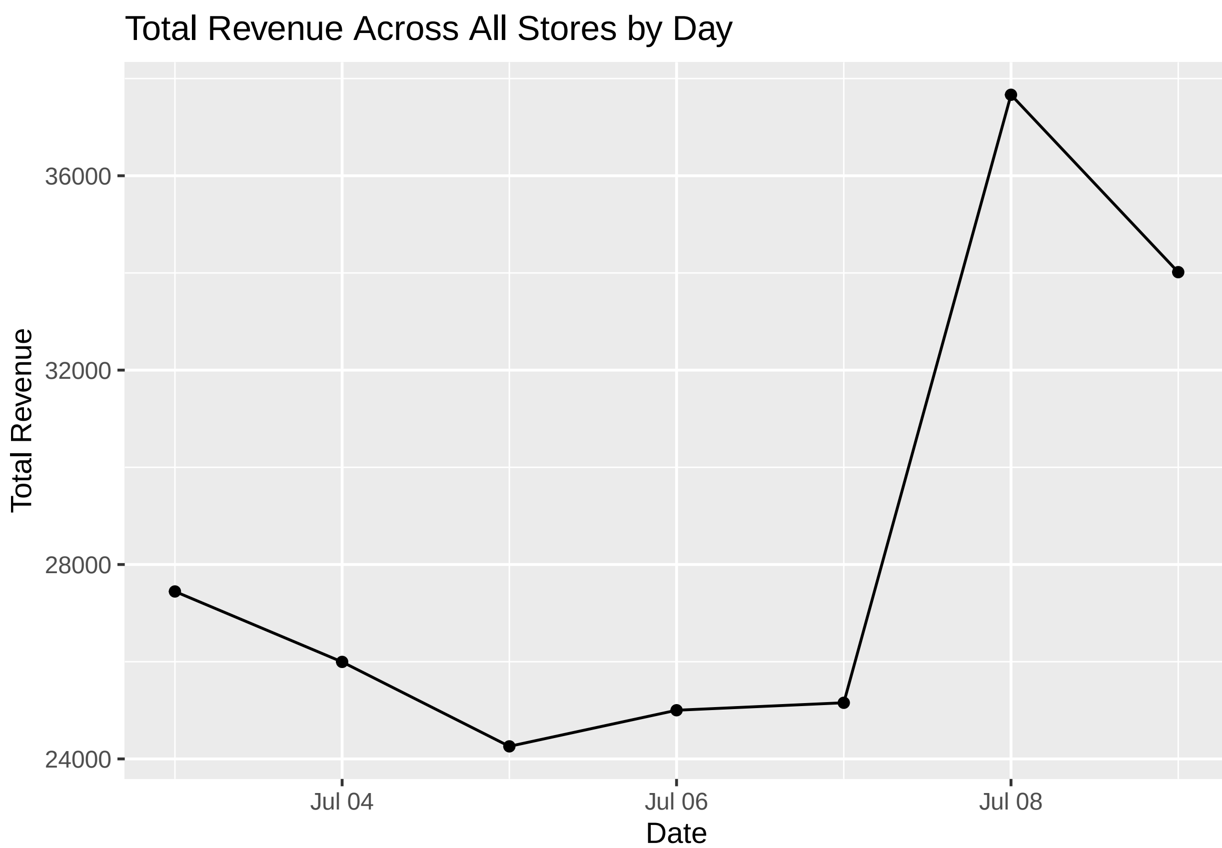
```r
# After grouping by store_id + date, there should be exactly ONE row per store-date.
stopifnot(nrow(store_daily_revenue) == nrow(distinct(sales, store_id, date)))
# Total revenue should not contain NA after summing with na.rm=TRUE
stopifnot(!any(is.na(store_daily_revenue$total_revenue)))

# Warn if any totals are negative (could indicate refunds/returns or data issues)
if (any(store_daily_revenue$total_revenue < 0, na.rm = TRUE)) {
warning("Some store_daily_revenue totals are negative - check source data.")
```

```
}
# Now we sum across stores to get ONE total for each day.
daily_total_revenue <- store_daily_revenue %>%
group_by(date) %>%
summarise(total_revenue_all_stores = sum(total_revenue), .groups = "drop") %>%
arrange(date)

nice_table(daily_total_revenue, "Total revenue across all stores by day")
```

Table 4: Total revenue across all stores by day

date	total_revenue_all_stores
2017-07-03	27445.55
2017-07-04	25995.74
2017-07-05	24257.92
2017-07-06	25001.26
2017-07-07	25154.68
2017-07-08	37666.13
2017-07-09	34016.19

```
#The line connects the days to show the pattern/trend over time.
#The points show the exact daily values.
ggplot(daily_total_revenue, aes(x = date, y = total_revenue_all_stores)) +
geom_line() +
geom_point() +
labs(
title = "Total Revenue Across All Stores by Day",
x = "Date",
y = "Total Revenue"
)
```




```r
# To compare stores across the whole week:
#   - sum each store's daily total revenue across all days.
store_weekly_revenue <- store_daily_revenue %>%
group_by(store_id) %>%
summarise(total_revenue_7days = sum(total_revenue), .groups = "drop") %>%
arrange(desc(total_revenue_7days))

nice_table(store_weekly_revenue, "Total revenue per store over 7 days")
```

Table 5: Total revenue per store over 7 days

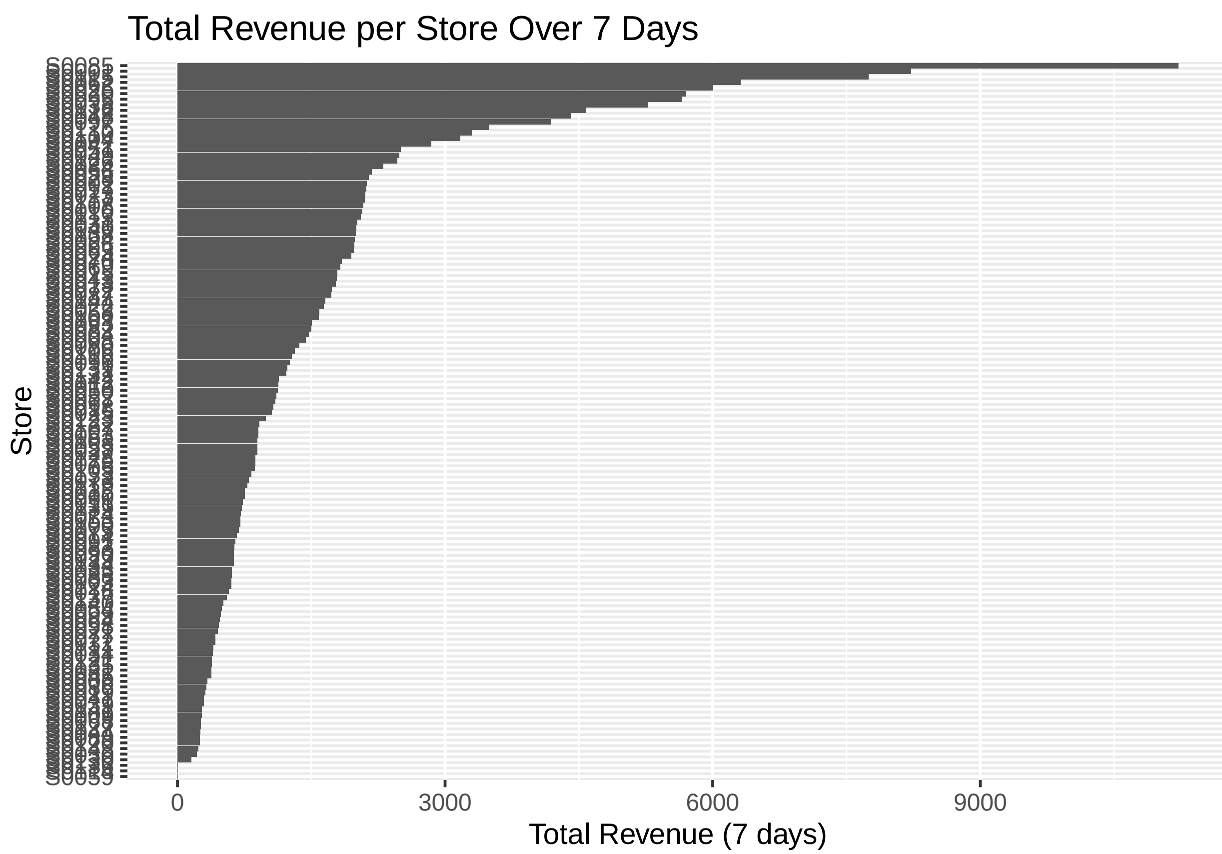
store_id	total_revenue_7days
S0085	11219.90
S0001	8224.19
S0115	7747.47
S0062	6311.25
S0026	6005.66
S0020	5698.97
S0095	5652.90
S0038	5276.60
S0112	4582.65
S0048	4407.80
S0097	4186.36

S0125	3495.60
S0110	3299.65
S0104	3165.98
S0042	2841.09
S0051	2502.01
S0049	2485.54
S0126	2462.68
S0084	2308.00
S0056	2175.47
S0028	2141.61
S0002	2122.74
S0017	2115.98
S0023	2104.75
S0117	2099.16
S0108	2081.87
S0010	2069.12
S0113	2053.27
S0031	2016.21
S0040	2005.14
S0138	1994.98
S0094	1984.15
S0066	1981.76
S0065	1976.88
S0024	1944.36
S0070	1841.93
S0069	1823.32
S0015	1790.93
S0043	1783.04
S0013	1771.00
S0072	1726.67
S0087	1723.94
S0144	1655.64
S0022	1639.26
S0058	1589.55
S0102	1585.25
S0093	1504.50
S0082	1498.39
S0004	1468.27
S0008	1439.65
S0080	1364.41
S0106	1316.14
S0116	1278.02
S0096	1257.64
S0131	1231.25
S0132	1221.13
S0142	1134.63
S0012	1131.57
S0050	1125.46
S0052	1108.60
S0067	1093.56

S0018	1071.38
S0025	1055.30
S0123	988.22
S0122	913.66
S0107	906.09
S0063	903.01
S0103	896.05
S0055	892.95
S0027	892.79
S0128	873.54
S0078	870.12
S0105	867.25
S0133	828.08
S0073	797.21
S0119	779.21
S0035	754.31
S0060	752.16
S0011	731.68
S0139	717.54
S0074	706.01
S0053	705.02
S0100	703.41
S0075	688.32
S0014	664.76
S0091	647.32
S0083	633.97
S0090	632.98
S0029	632.19
S0134	629.92
S0033	609.04
S0088	606.21
S0003	603.76
S0124	600.91
S0016	576.33
S0137	552.60
S0140	510.46
S0054	495.05
S0099	486.19
S0064	471.83
S0098	463.02
S0021	452.85
S0077	422.87
S0032	421.27
S0111	399.31
S0034	393.89
S0121	382.78
S0135	381.73
S0081	379.32
S0045	378.45
S0006	334.99

S0086	319.92
S0019	308.32
S0041	294.35
S0039	294.29
S0141	274.12
S0009	270.10
S0068	259.69
S0127	259.43
S0044	257.31
S0089	248.30
S0120	246.95
S0143	232.93
S0030	214.07
S0130	154.88
S0059	0.00
S0114	0.00
S0136	0.00

```
# Plot weekly revenue per store
ggplot(store_weekly_revenue, aes(x = reorder(store_id, total_revenue_7days), y = total_revenue_7days)) +
geom_col() +
coord_flip() +
labs(
title = "Total Revenue per Store Over 7 Days",
x = "Store",
y = "Total Revenue (7 days)"
)
```



best_day <- daily_total_revenue %>% slice_max(total_revenue_all_stores, n = 1, with_ties = FALSE)
worst_day <- daily_total_revenue %>% slice_min(total_revenue_all_stores, n = 1, with_ties = FALSE)
top_store <- store_weekly_revenue %>% slice_max(total_revenue_7days, n = 1, with_ties = FALSE)

The highest total revenue day was **2017-07-08** with 3.766613×10^4 revenue.
The lowest total revenue day was **2017-07-05** with 2.425792×10^4 revenue.
The top store over 7 days was **S0085** with 1.12199×10^4 revenue.

==

TASK 2: Most popular product type (Hierarchy 1)

==

"Most popular" is measured using total sales quantity. Sales is joined with product hierarchy, then aggregated by hierarchy1_id. The ranked table includes number of hierarchy2 subtypes, product count, total quantity sold, and revenue.

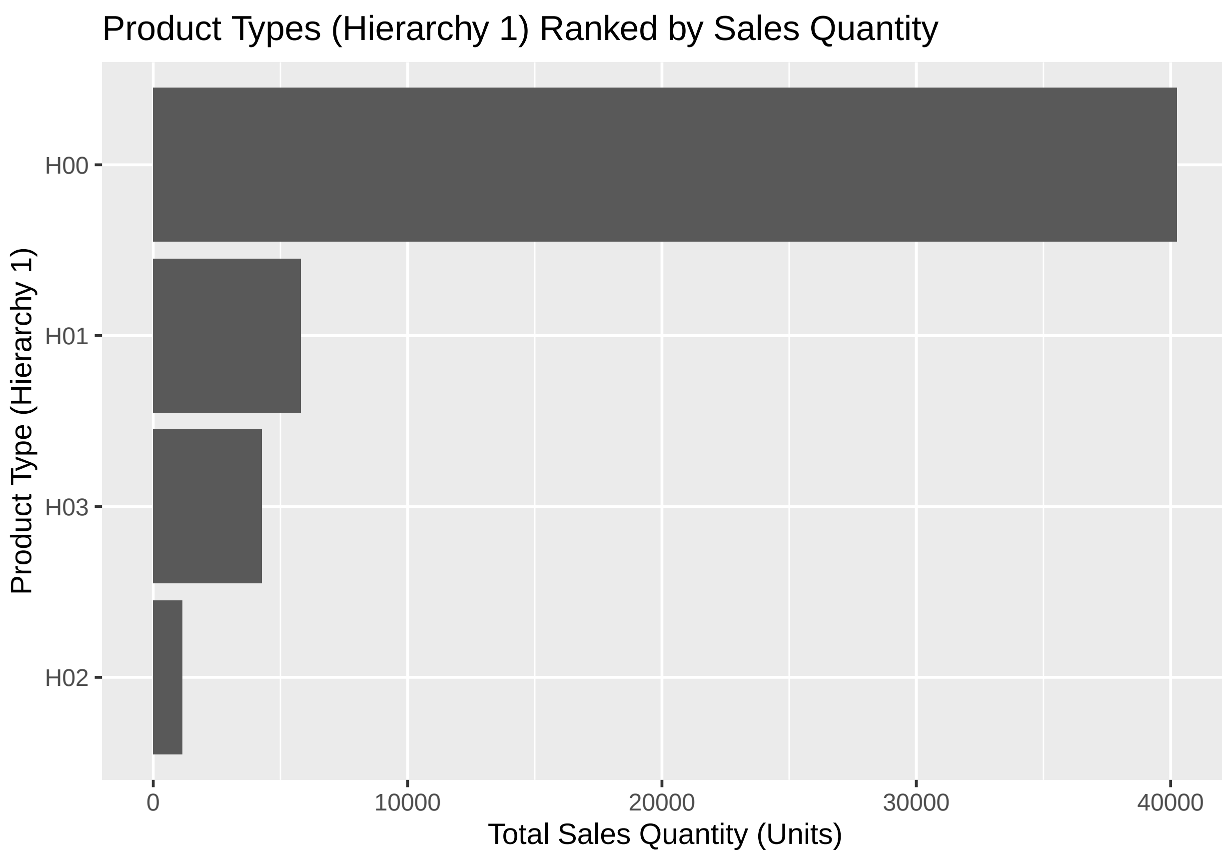
```
# left_join keeps all sales rows and adds hierarchy columns using product_id
sales_with_hierarchy <- sales %>%
left_join(product_hierarchy, by = "product_id")

# Validation: join should not change the number of sales rows
stopifnot(nrow(sales_with_hierarchy) == nrow(sales))
```

```r
product_type_ranked <- sales_with_hierarchy %>%
group_by(hierarchy1_id) %>%
summarise(
subtypes_h2 = n_distinct(hierarchy2_id),
products = n_distinct(product_id),
sales_qty = sum(sales, na.rm = TRUE),
revenue = sum(revenue, na.rm = TRUE),
.groups = "drop"
) %>%
arrange(desc(sales_qty))

nice_table(product_type_ranked, "Product types ranked by total sales quantity")
```

Table 6: Product types ranked by total sales quantity

hierarchy1_id	subtypes_h2	products	sales_qty	revenue
H00	5	128	40256.82	100165.44
H01	4	99	5797.00	61773.15
H03	7	119	4266.00	25377.66
H02	2	2	1141.98	12221.22

```r
top1 <- product_type_ranked %>% slice(1)
top2 <- product_type_ranked %>% slice(2)

# Plot ranked product types
ggplot(product_type_ranked, aes(x = reorder(hierarchy1_id, sales_qty), y = sales_qty)) +
geom_col() +
coord_flip() +
labs(
title = "Product Types (Hierarchy 1) Ranked by Sales Quantity",
x = "Product Type (Hierarchy 1)",
y = "Total Sales Quantity (Units)"
)
```




```
#"This quickly shows which product category is most popular by units sold."
#"The product type with the largest bar is the most popular because it sold the most units."

#"This is volume popularity, not revenue popularity."
```

Most popular hierarchy1 is **H00** (sales quantity 4.0256818×10^4, revenue 1.0016544×10^5).
Second most popular is **H01** (sales quantity **5797**, revenue 6.177315×10^4).

==

TASK 3: Store types and store size vs revenue

==

Store metadata is joined to sales. The two most common store types are identified by number of stores. Weekly sales volume and revenue are compared. Store size vs revenue is tested using correlation and visualised using a scatter plot.

```
sales_with_store <- sales %>%
left_join(store_cities, by = "store_id")

# Validation: join should not change number of sales rows
```

```r
stopifnot(nrow(sales_with_store) == nrow(sales))

storetype_counts <- store_cities %>%
group_by(storetype_id) %>%
summarise(num_stores = n_distinct(store_id), .groups = "drop") %>%
arrange(desc(num_stores))

nice_table(storetype_counts, "Store type counts (by number of stores)")
```

Table 7: Store type counts (by number of stores)

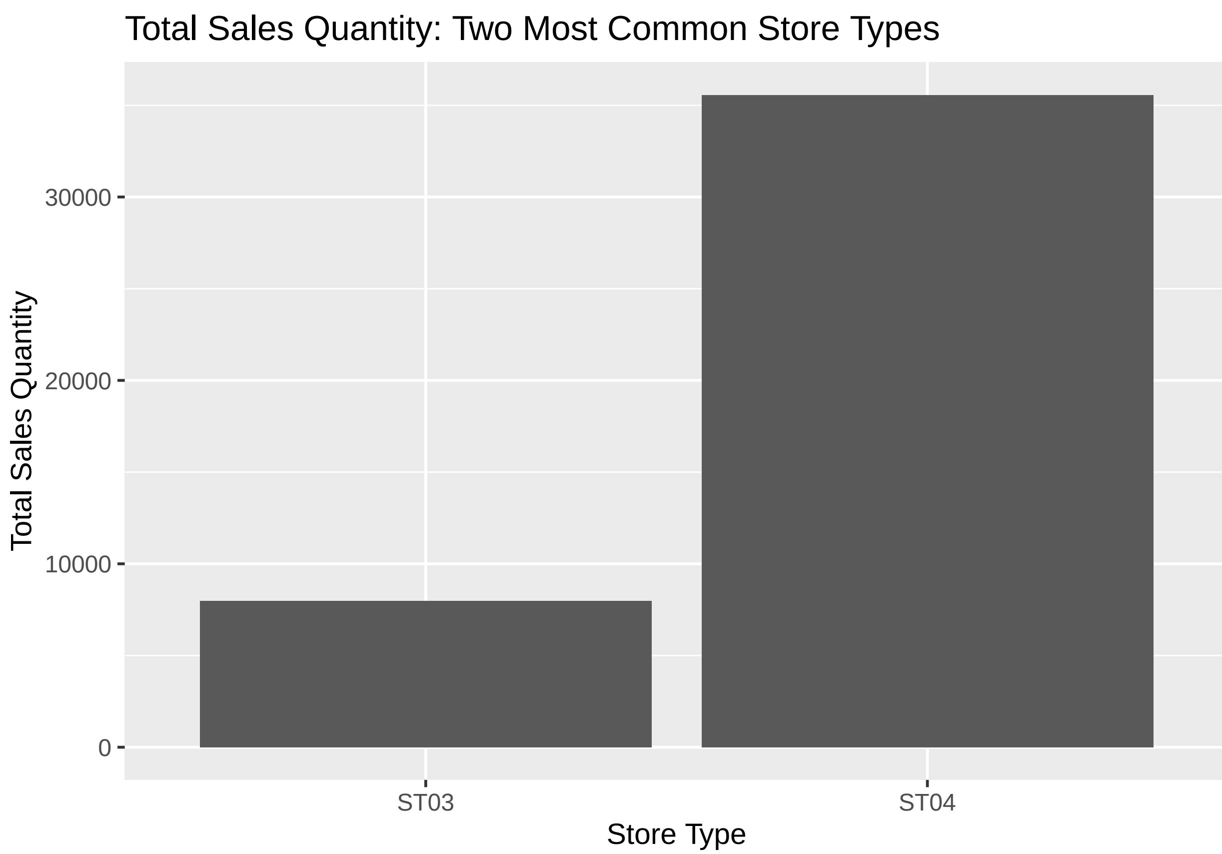
storetype_id	num_stores
ST04	83
ST03	53
ST01	4
ST02	4

```r
# Top 2 store types
top_two_storetypes <- storetype_counts %>% slice(1:2) %>% pull(storetype_id)

storetype_weekly <- sales_with_store %>%
filter(storetype_id %in% top_two_storetypes) %>%
group_by(storetype_id) %>%
summarise(
total_sales_qty = sum(sales, na.rm = TRUE),
total_revenue = sum(revenue, na.rm = TRUE),
num_stores = n_distinct(store_id),
.groups = "drop"
) %>%
arrange(desc(total_sales_qty))

nice_table(storetype_weekly, "Sales and revenue for the two most common store types")
```

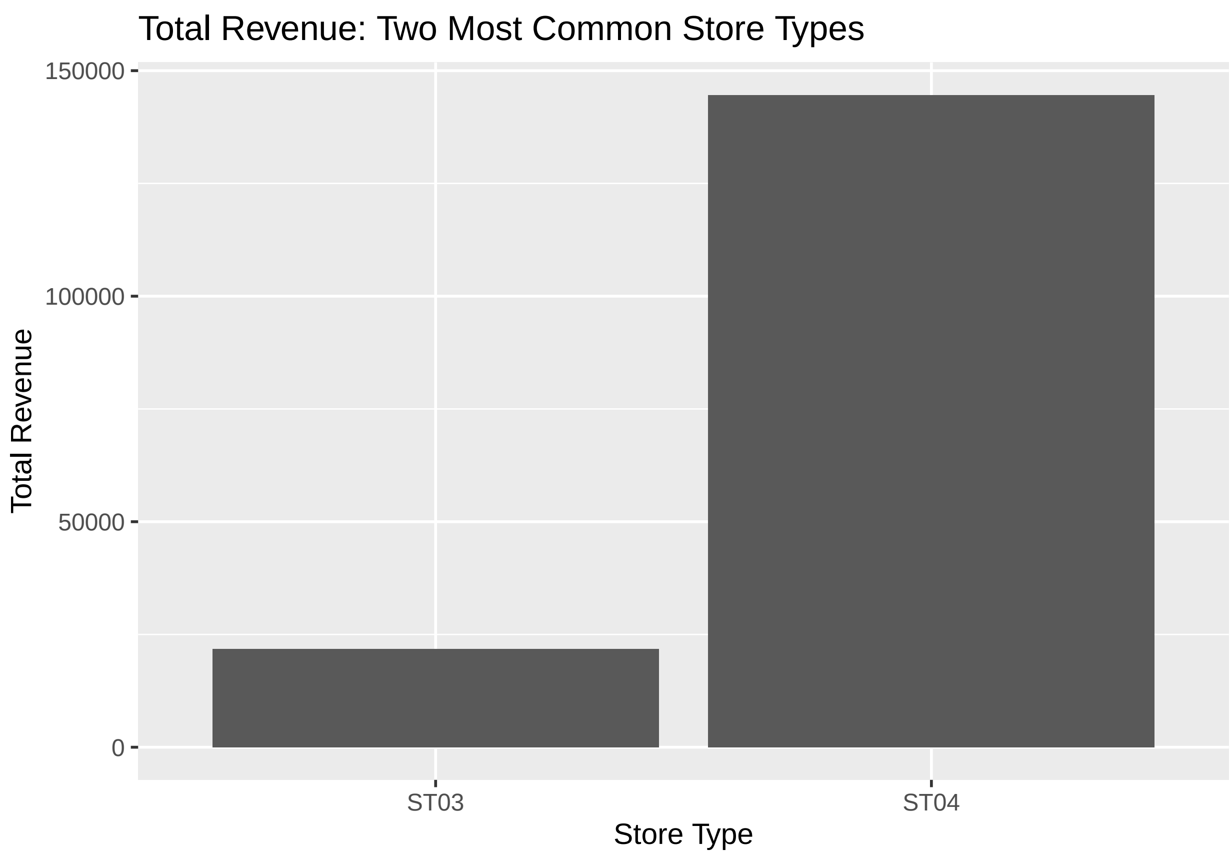
Table 8: Sales and revenue for the two most common store types

storetype_id	total_sales_qty	total_revenue	num_stores
ST04	35566.55	144628.73	73
ST03	7980.01	21776.75	47

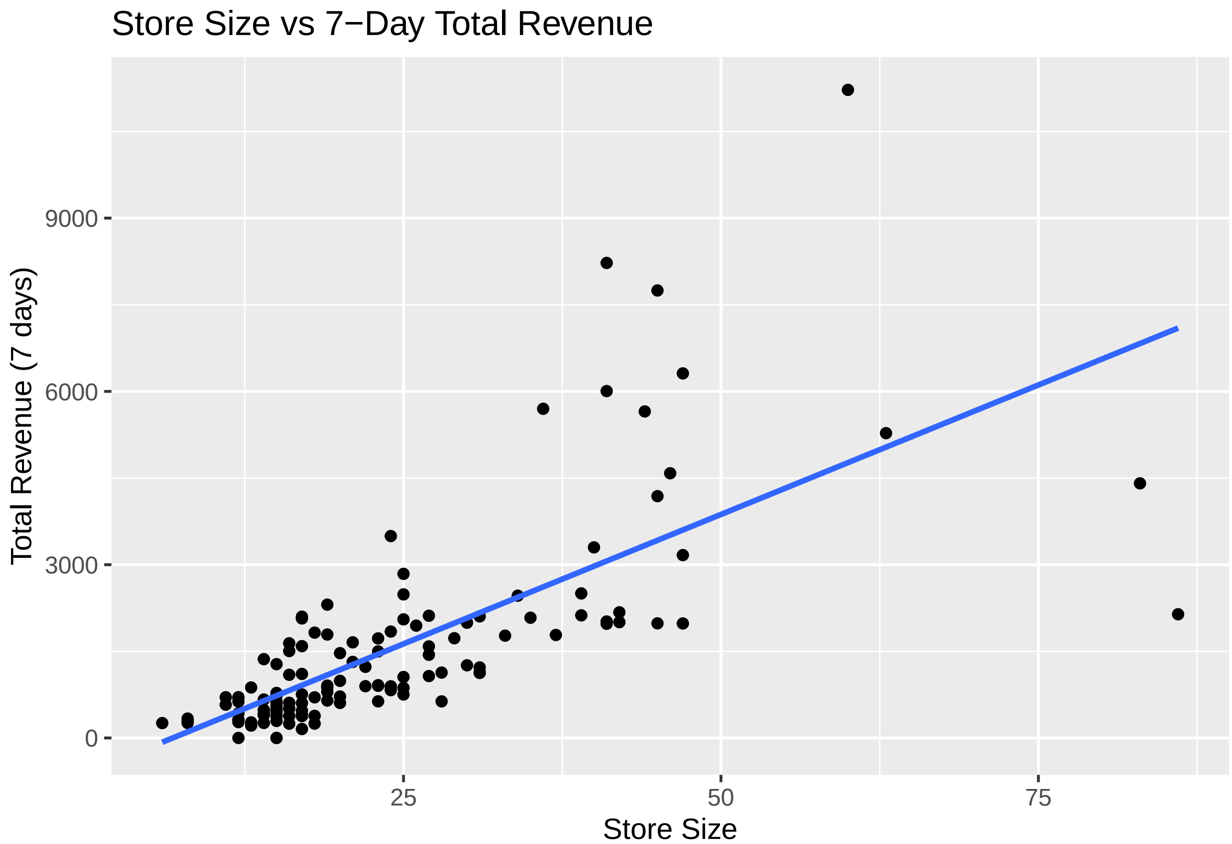
```r
# Plot total sales quantity
#"Choosing the two most common store types gives a fairer comparison because both have enough stores an
#If one bar is higher: that store type has higher total sales volume.
ggplot(storetype_weekly, aes(x = storetype_id, y = total_sales_qty)) +
geom_col() +
labs(
title = "Total Sales Quantity: Two Most Common Store Types",
x = "Store Type",
y = "Total Sales Quantity"
)
```




```r
# Plot total revenue
#"This reveals which store type generates more income overall."
ggplot(storetype_weekly, aes(x = storetype_id, y = total_revenue)) +
geom_col() +
labs(
title = "Total Revenue: Two Most Common Store Types",
x = "Store Type",
y = "Total Revenue"
)
```




```r
# We want one row per store, so we group by store_id and store_size,
# then sum revenue across the week.
store_weekly_perf <- sales_with_store %>%
group_by(store_id, storetype_id, store_size) %>%
summarise(total_revenue_7days = sum(revenue, na.rm = TRUE), .groups = "drop")

# Correlation measures strength of linear relationship
cor_value <- cor(store_weekly_perf$store_size, store_weekly_perf$total_revenue_7days, use = "complete.ol

# Scatter plot with regression line
#If the line slopes upward: larger stores tend to make more revenue.
#If points are close to the line: relationship is strong.
#If points are widely scattered: relationship is weak.
ggplot(store_weekly_perf, aes(x = store_size, y = total_revenue_7days)) +
geom_point() +
geom_smooth(method = "lm", se = FALSE) +
labs(
title = "Store Size vs 7-Day Total Revenue",
x = "Store Size",
y = "Total Revenue (7 days)"
)
```




```
#"The correlation value supports this visual relationship, showing whether size and revenue are strongl
```

The correlation between store size and 7-day revenue is **0.701**.

==

TASK 4: Promotion levels and effectiveness

==

Promotion types and levels are listed first. Promotion effectiveness is assessed by comparing average sales when promotions are applied versus not applied, then examining sales patterns across promotion bins and discount rates.

Table 9: Promotion type 1 levels used

promo_type_1	promo_bin_1
PR03	verylow
PR05	high
PR05	low
PR05	moderate
PR05	verylow

PR06	low
PR06	verylow
PR08	veryhigh
PR09	high
PR09	low
PR10	verylow
PR12	veryhigh
PR12	verylow
PR13	verylow
PR14	

Table 10: Promotion discount 2 levels used

promo_discount_type_2	promo_discount_2
NA	NA

Table 11: Average sales/revenue: promo type 1 vs no promo

has_promo1	avg_sales	avg_revenue	n
TRUE	0.49	1.92	104000

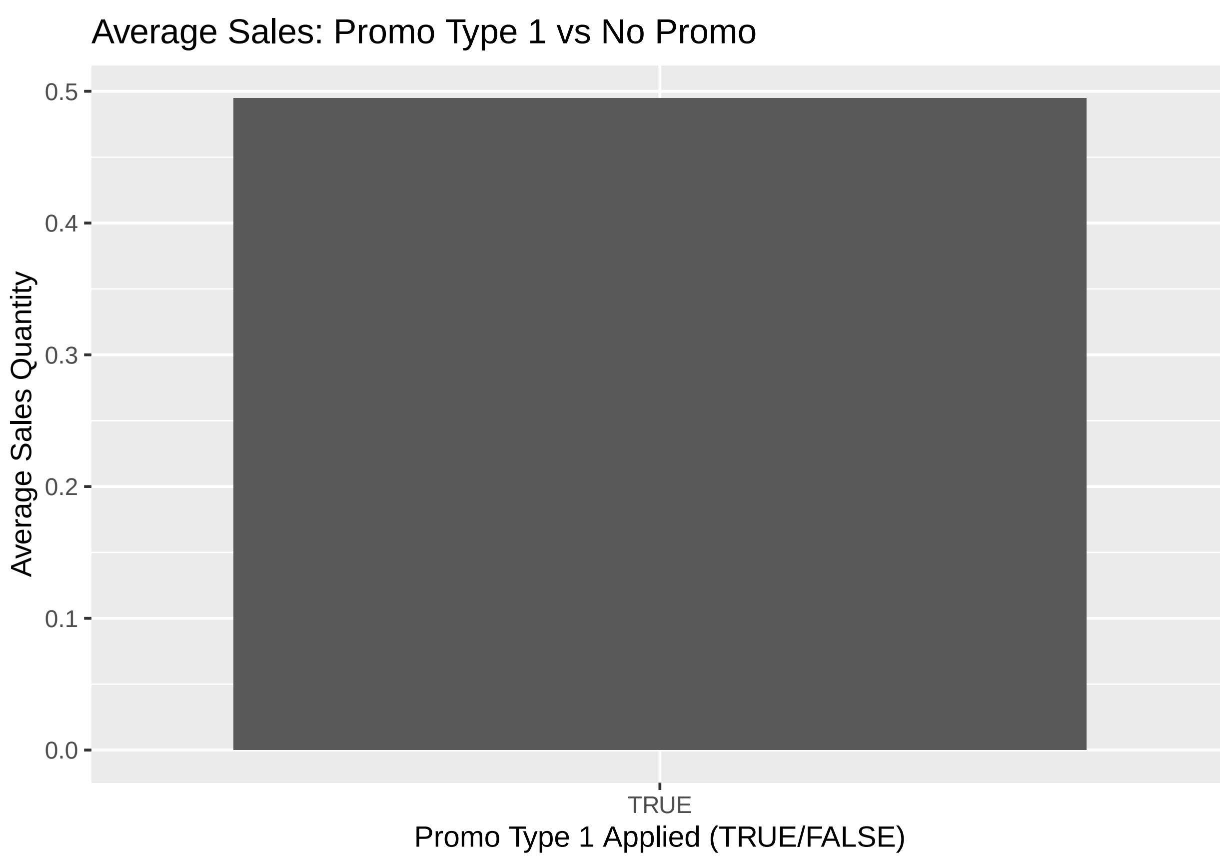


Table 12: Average sales by promo type and promo level bin

promo_type_1	promo_bin_1	avg_sales	avg_revenue	n
PR03	verylow	0.63	3.12	286
PR05	high	0.67	1.50	123
PR05	low	0.16	2.63	744
PR05	moderate	0.43	6.39	14
PR05	verylow	0.13	1.79	240
PR06	low	0.01	0.04	175
PR06	verylow	0.09	0.45	481
PR08	veryhigh	5.13	87.13	126
PR09	high	0.56	3.01	190
PR09	low	0.73	1.47	1638
PR10	verylow	1.10	28.58	58
PR12	veryhigh	0.84	1.05	3196
PR12	verylow	0.28	2.40	1804
PR13	verylow	5.04	98.04	26
PR14		0.48	1.79	94899

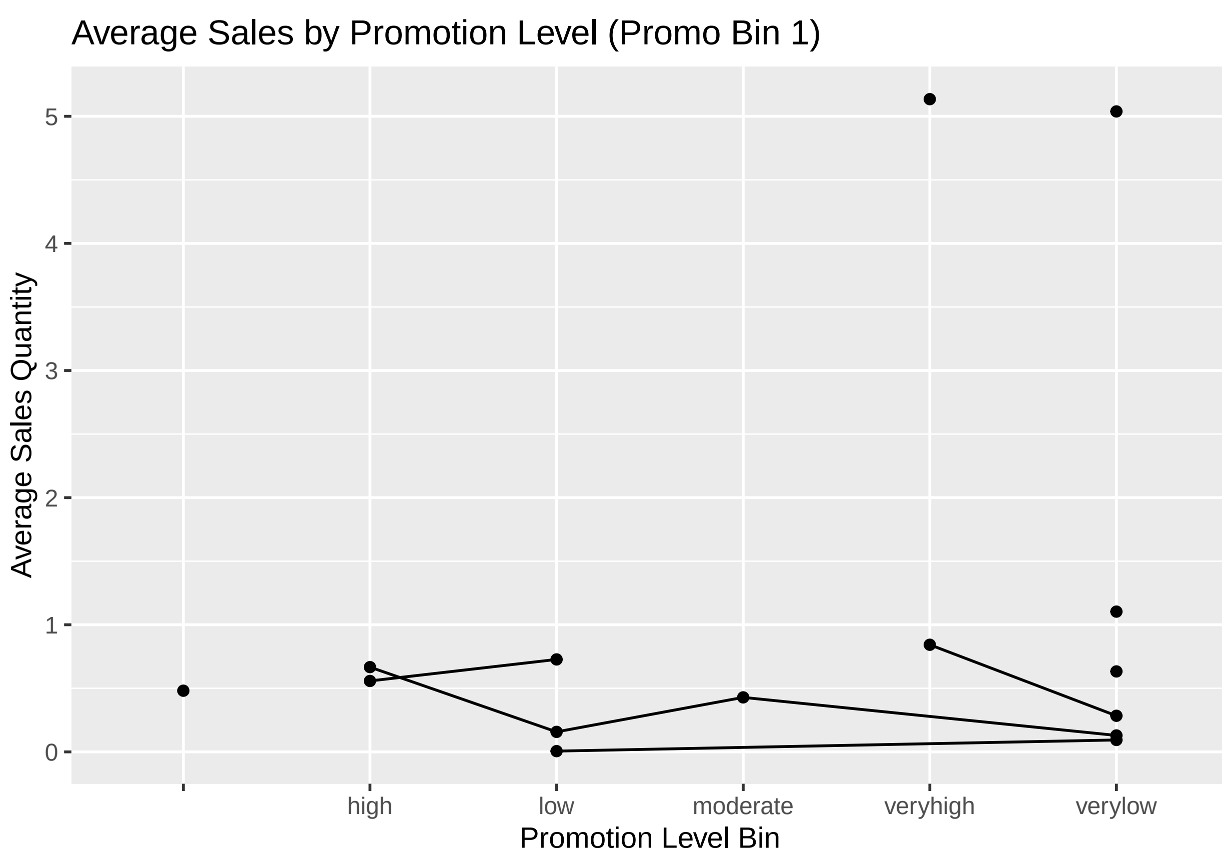


Table 13: Average sales by discount type and discount rate

promo_discount_type_2	promo_discount_2_num	avg_sales	avg_revenue	n
NA	NA	NA	NA	NA

:———————— ————————: ————: —————: —:

```
## No non-zero Promo Discount 2 values were used during the week, so a discount-rate effectiveness plot
```