Project 1

Parking Citation Data Integration with Pentaho and PostgreSQL Data Warehouse

Introduction

This project converts the Los Angeles Parking Citations CSV (Kaggle / City of LA) into a clean, query-ready warehouse using Pentaho Data Integration (PDI). We built an end-to-end ETL flow that reads the raw file, keeps only relevant columns, removes rows with missing/invalid values, standardizes types, and then loads a star schema in PostgreSQL. The star schema has four dimensions—Date, Violation, Geo, Vehicle—and one Fact table that records each citation with its keys.

Objectives

- Turn a large raw CSV into clean, consistent tables that are easy to query.
- Design a simple star schema with Date/Violation/Geo/Vehicle dimensions and one central Fact table.
- Drop/standardize null or invalid fields so keys and joins are reliable.
- Use PDI steps (Sort, Lookups, Table Output) to load dimensions first, then the fact.
- Use a Pentaho Job to orchestrate tasks, define error paths, and send notifications.
- Produce a warehouse that supports basic questions (by date, location, vehicle, violation) and can be extended later for BI or modeling.

Project Overflow & Implementation

Dataset Description

We use the Los Angeles Parking Citations file published by the City of Los Angeles (Socrata) and mirrored on Kaggle as parking-citations.csv. It is a single, ticket-level CSV containing about 1.23 million rows and 22 columns; each row represents one issued citation.

The columns fall into four groups. Date/time fields record when the ticket was written (Issue Date and Issue Time). Violation fields hold the official violation code and its description. Vehicle fields describe the cited vehicle, such as plate state, body style, make, and color (with some helper description fields). Location fields store the street text (Location), a Route code used by the city, and XY/lat-long coordinates supplied by the publisher.

Data quality varies across fields. The dataset uses placeholders like 99999 in latitude/longitude when a coordinate is missing, and some administrative columns are empty or repeated. For this project we kept the useful, non-null columns from those 22 fields, standardized types (dates, codes, and text casing), and treated 99999 coordinates as missing. The cleaned data then feeds our warehouse model: Date, Violation, Geo, and Vehicle dimensions, with one fact row per ticket to support analysis by time, place, vehicle, and violation.

Source: City of Los Angeles Open Data (Socrata) via Kaggle — Los Angeles Parking Citations.

Extraction Process

Input

We start ETL with CSV File Input in Pentaho Data Integration.
It reads parking_citations.csv (comma delimiter, " enclosure, header row present, lazy conversion on) and streams the raw 1.23M × 22 data into the pipeline for cleaning.

Fig. 1. Extraction step: CSV File Input

We used Select Values to keep only the needed fields and remove columns with many nulls, noise, or irrelevant information. This simplified the data before building the dimensions and fact table.

Filter Row

We applied Filter Rows to keep only valid records. Condition: key fields must not be NULL (Issue Date, Plate State, Body Style, Location, Route, Violation Code/Description) and numeric checks (serial > 0; optional fields like amounts > 0 if used). Valid rows go to selected row; failures go to remove row. This removes missing/invalid data before loading.

Fig.2. Filter Row

Sort Row

We sorted rows by serial (ascending) to keep a stable, consistent order. This avoids messy/random ordering, helps detect duplicates, and prepares the stream for later joins (Multiway Merge Join needs sorted inputs).

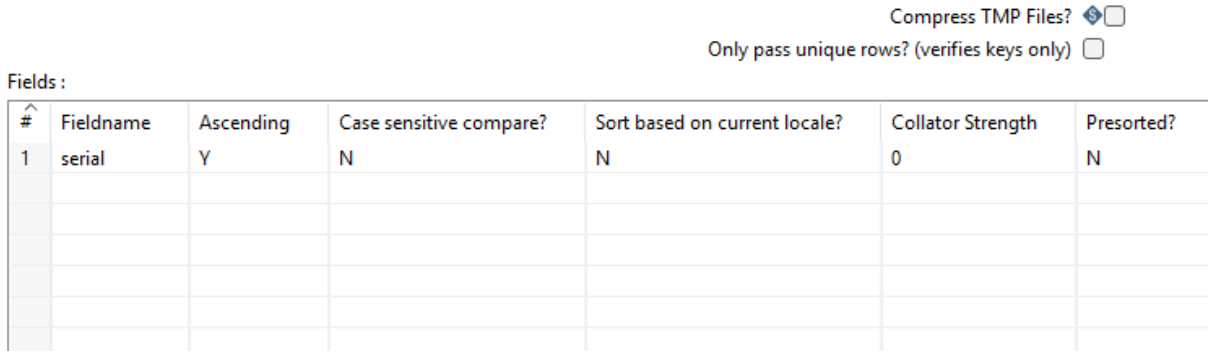


Fig.3. Sort Row by serial

Output

In this step, we saved the Fresh Data to our computer as a text file using Text File Output.

serial	issue date	rp state plate	body style	location	route	violation code	violation description	fine amount
1	2015-12-21	CA	PA	13147 WELBY WAY	1521	4000A1	NO EVIDENCE OF REG	50
2	2015-12-21	CA	VN	525 S MAIN ST	1C51	4000A1	NO EVIDENCE OF REG	50
3	2015-12-21	CA	PA	200 WORLD WAY	2R2	8939	WHITE CURB	58
5	2015-09-15	CA	PA	GEORGIA ST/OLYMPIC	1FB70	8069A	NO STOPPING/STANDING	93
6	2015-09-15	CA	VN	SAN PEDRO S/O BOYD	1A35W	4000A1	NO EVIDENCE OF REG	50

Fig. 4. Fresh Data :Text File Output

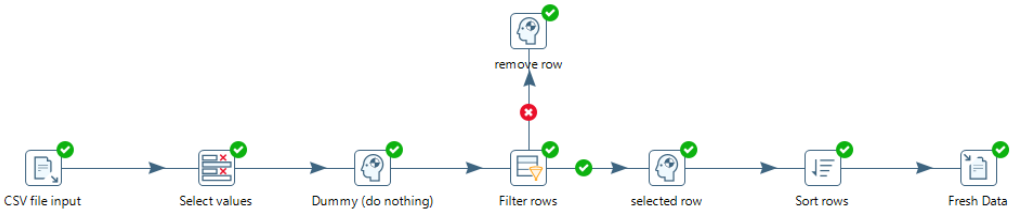


Fig.5. Extraction Process

Dimention & Fact Table

Date Dimention

From Fresh Data, we keep issue date and serial, sort by serial, then split the date into parts (year, month, day) and load to the warehouse as the Date table (date_dim) using Date Table Output.

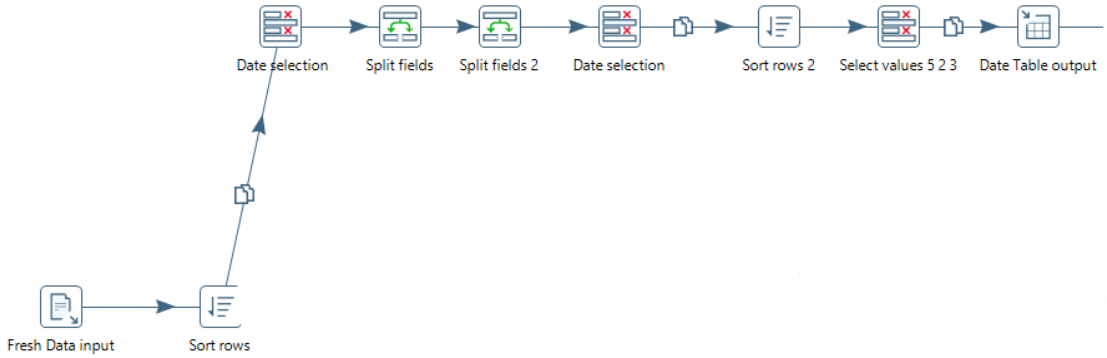


Fig.6. Date Dimention

Day/Month/Year Split

We used Split fields on issue date with the - delimiter to create three new fields: year, month, and day.

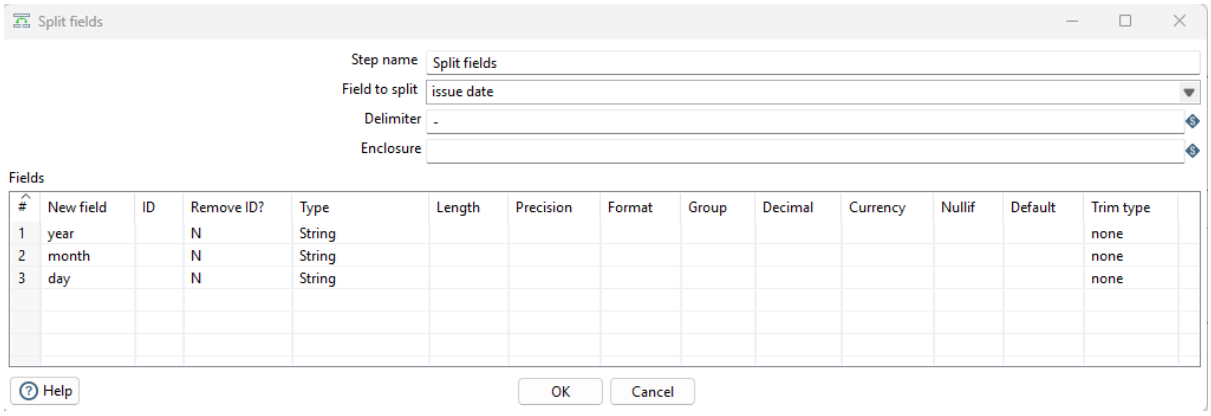
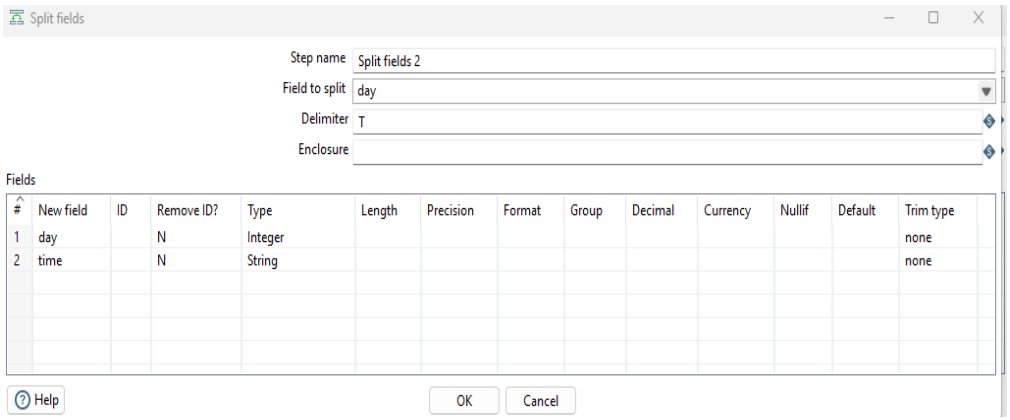


Fig.7. Date Split

Day Split

We used a second Split fields on the day value with delimiter “T” to separate the date from the timestamp. We keep the left part (pure date) and drop the right part (00:00:00.000), removing the “T00…”.



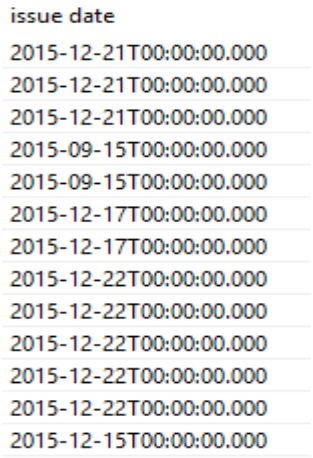
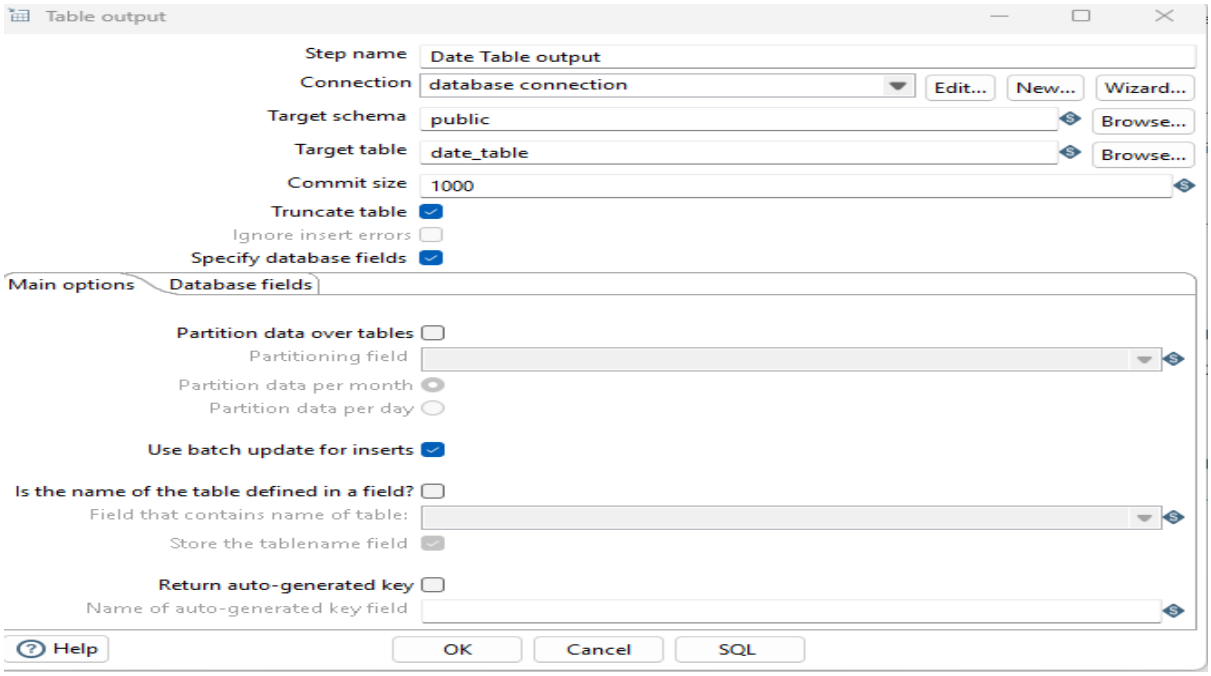


Fig.8. Day Split

Date Table Output

This is the Table Output step that sends the date-dimension data to the PostgreSQL warehouse (public.date_table), using batch inserts and truncating the table before each load.



Violation Dimention

From Fresh Data, we sort by serial, keep only violation code and violation description (violation selection → Select Values), and load them to the warehouse as Dim_Violation using the Violation Table Output.

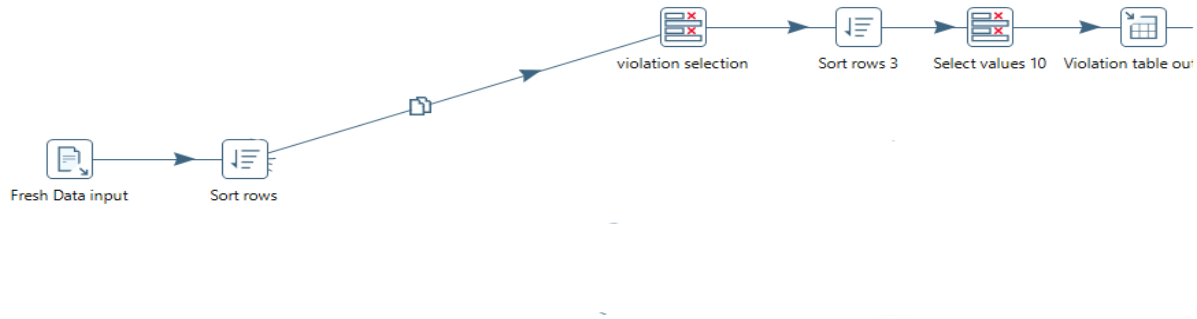


Fig.9. Violation Dimention

Geo Dimention

From Fresh Data, we sort by serial, pick the geo fields (location, route), and load them to the warehouse as Dim_Geo using Geo Table Output.

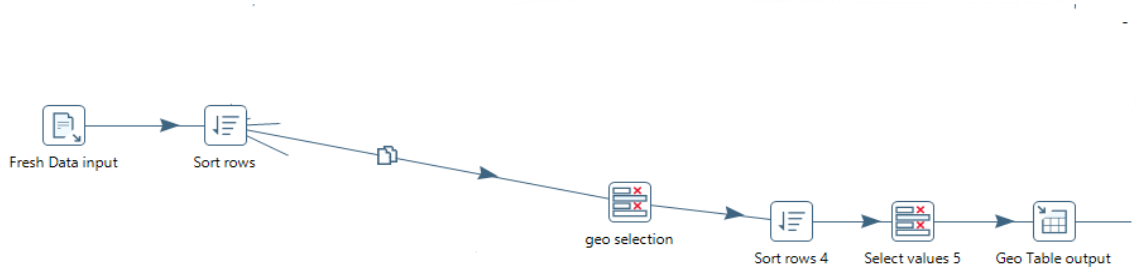


Fig.10. Geo Dimention

Vehicle Dimention

From Fresh Data, we sort by serial, then select the vehicle fields RP State Plate and Body Style. We send these to the warehouse as Dim_Vehicle using the Vehicle Table Output.

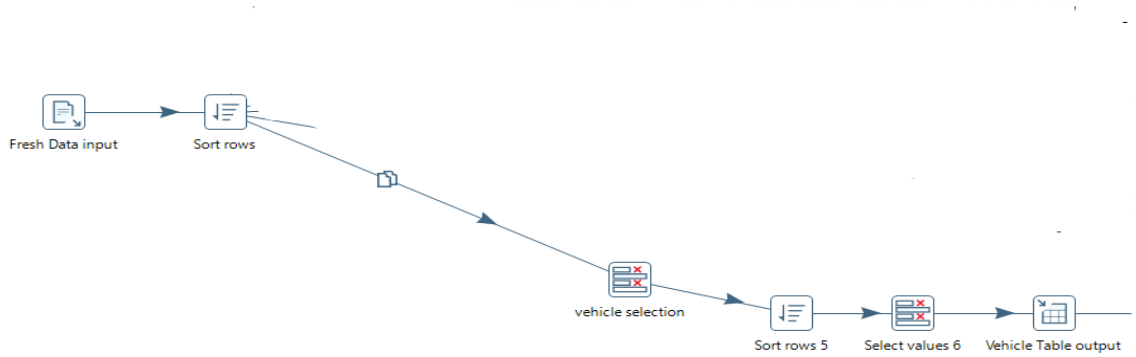


Fig.11. Vehicle Dimention

Fact Table

For the fact table, we use Database Lookup to fetch each dimension key from the dim tables (date_id, violation_id, geo_id, vehic_id). Then we Multiway Merge Join those keys with serial (fact PK) and fine_amount, and load the result into the Fact table in the warehouse.

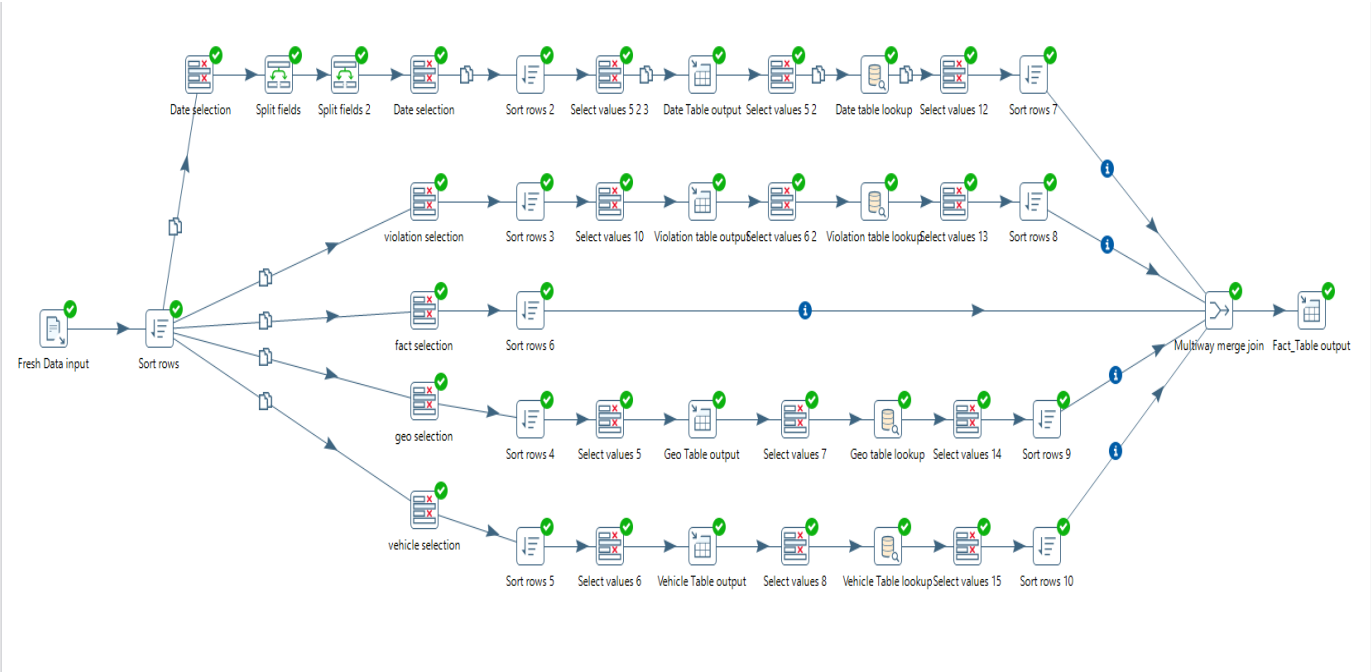


Fig. 12. Database Lookups + Multiway Merge Join to build the Fact table Multi Join

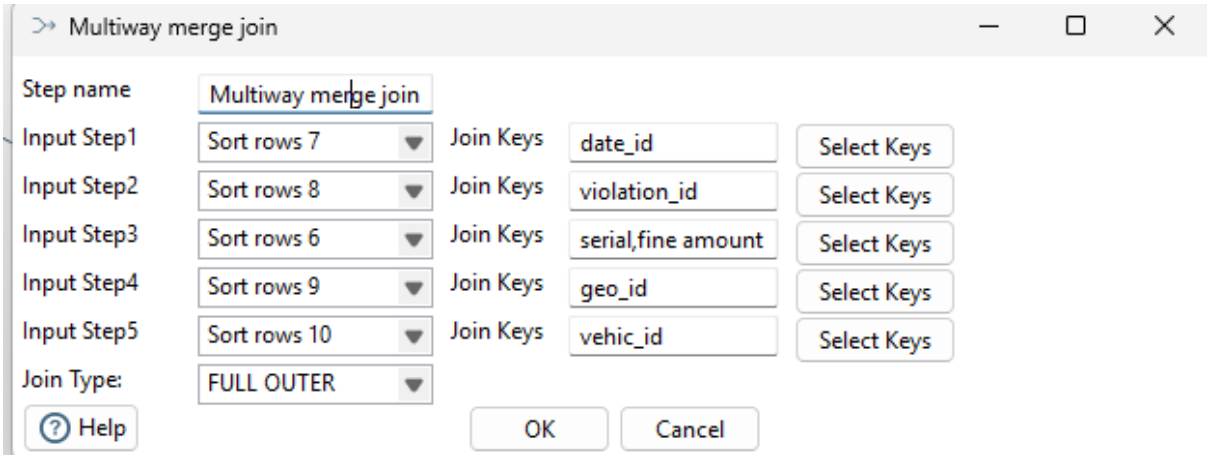


Fig.13. Dimention and Fact table

Fact Table Preview

Execution Results

Logging | Execution History | Step Metrics | Performance Graph | Metrics | Preview data

First rows | Last rows | Off

#	date_id	violation_id	serial	fine amount	geo_id	vehic_id
1	1	1	1	50	1	1
2	2	2	2	50	2	2
3	3	3	3	58	3	3
4	5	5	5	93	5	5
5	6	6	6	50	6	6
6	7	7	7	163	7	7
7	8	8	8	163	8	8
8	9	9	9	93	9	9
9	10	10	10	93	10	10
10	11	11	11	93	11	11
11	12	12	12	93	12	12
12	13	13	13	50	13	13
13	14	14	14	93	14	14
14	18	18	18	50	18	18
15	19	19	19	93	19	19
16	20	20	20	73	20	20
17	22	22	22	50	22	22
18	23	23	23	50	23	23
19	24	24	24	50	24	24
20	25	25	25	25	25	25
21	26	26	26	25	26	26
22	27	27	27	25	27	27
23	28	28	28	25	28	28
24	29	29	29	25	29	29
25	31	31	31	25	31	31
26	32	32	32	50	32	32
27	33	33	33	50	33	33
28	34	34	34	93	34	34
29	35	35	35	50	35	35
30	36	36	36	50	36	36
31	37	37	37	50	37	37
32	38	38	38	63	38	38

Fig.14. Fact table Preview

ETL Job

This job controls the whole run. It starts, executes Extraction Process, then Dimension & Fact Table Process. If both finish OK, it sends a Success Mail. If any step fails, the red error hops trigger Abort job to stop the run.

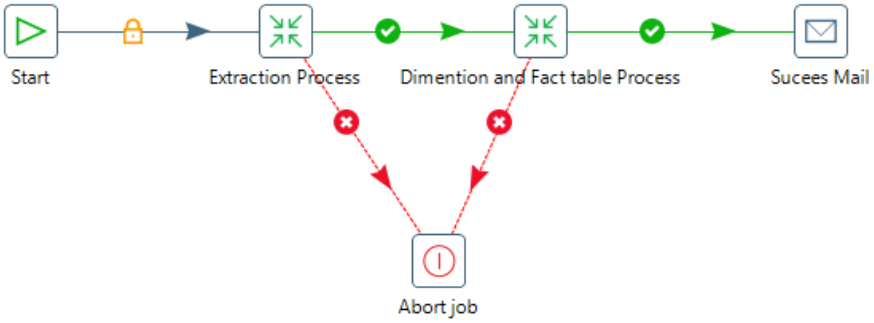


Fig.15. Job Orchestration

We used: Gmail SMTP with TLS security on port 587, with authentication (email + password/app-password).

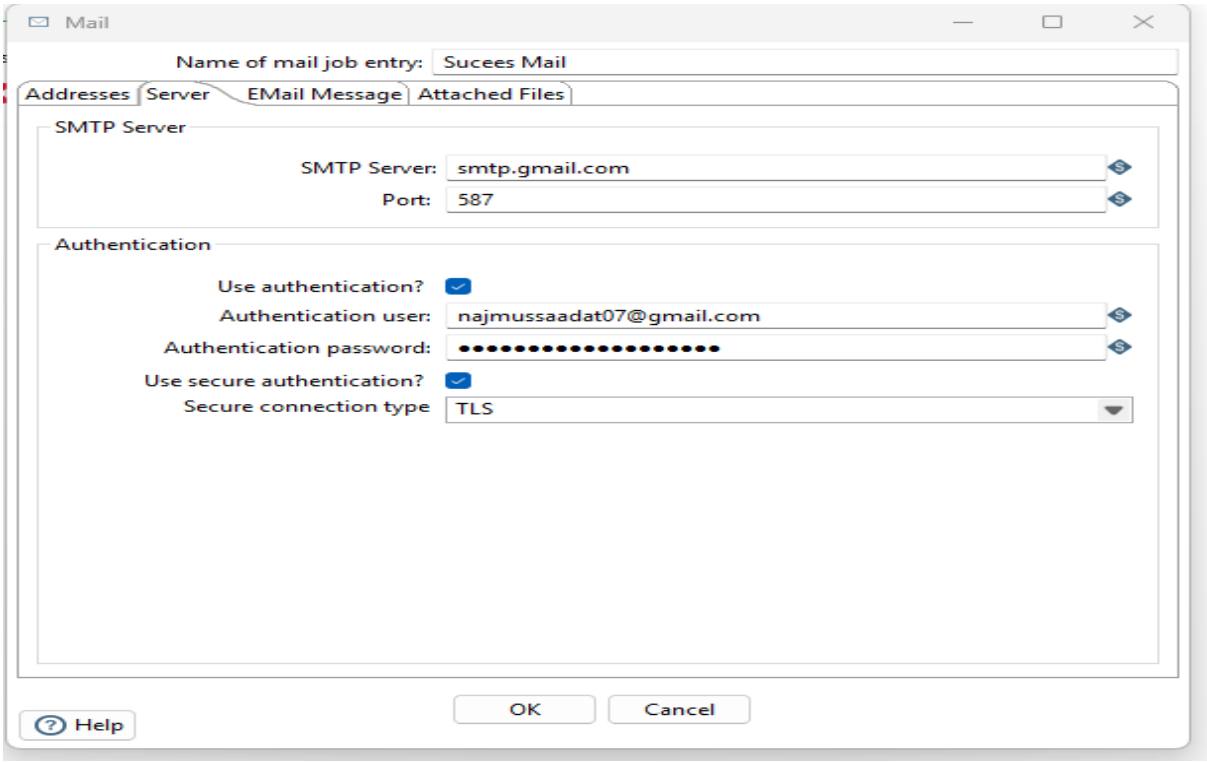


Fig.16. Gmail SMTP with authentication

PostgreSQL Data Warehouse — Fact Table Overview

This view shows our warehouse schema (public) with the four dimensions and the fact_table. The query window selects key columns from the fact: serial (PK) plus foreign keys date_id, violation_id, geo_id, vehic_id and the fine_amount measure. The data output confirms the load succeeded with about 1,037,080 rows in the fact table.

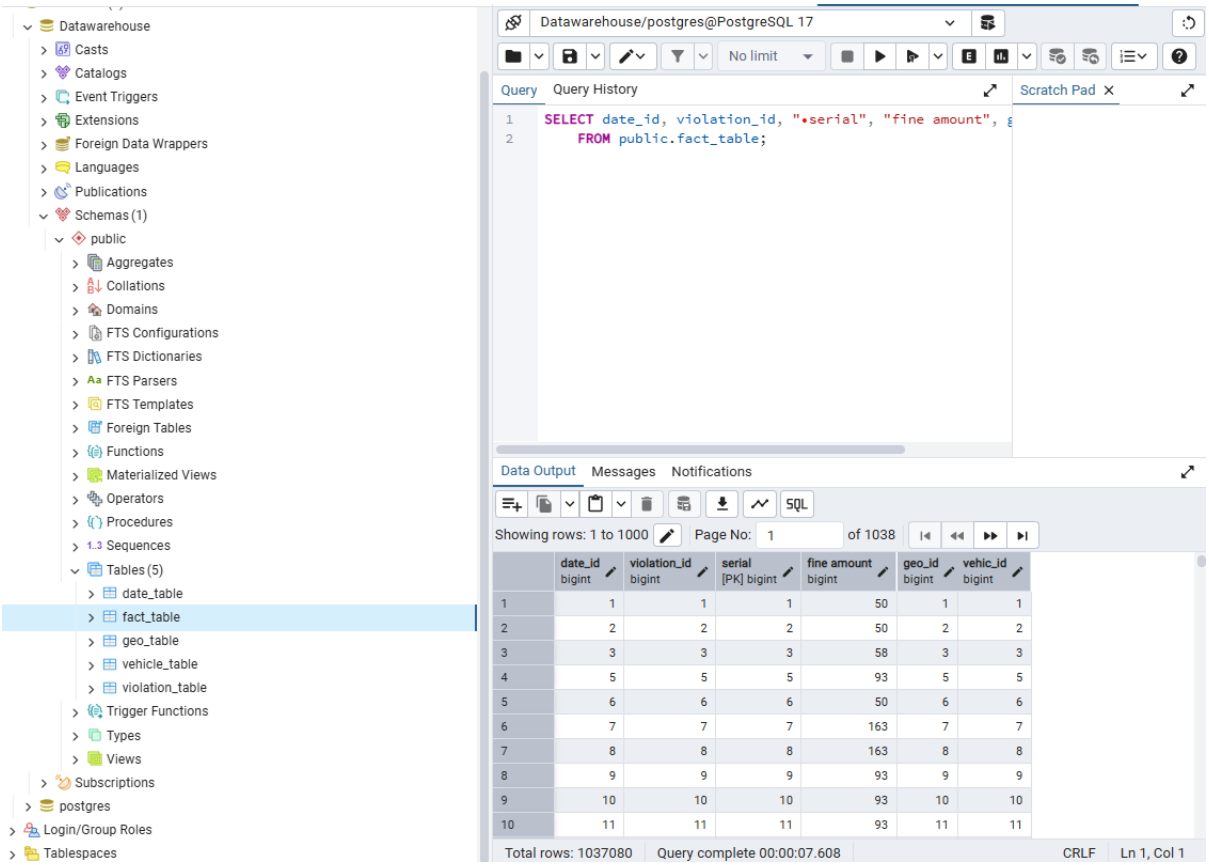
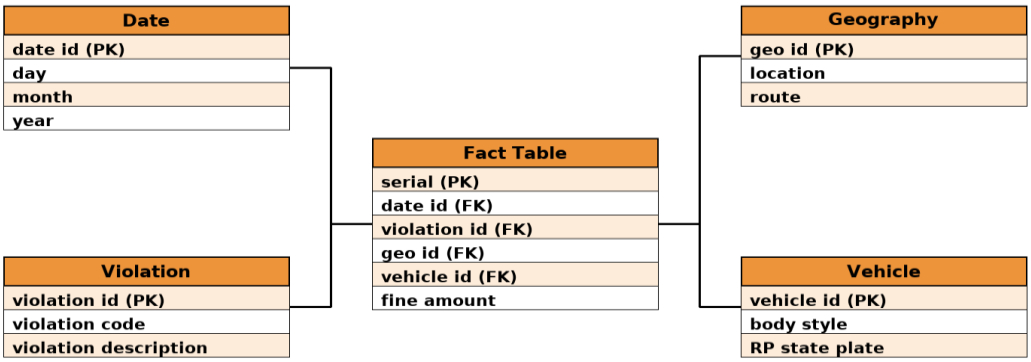


Fig. 17. PostgreSQL Data Warehouse — Fact table in pgAdmin

ER Diagram

This is a star schema: one central Fact_Citation table (one row = one serial number) linked to four dimensions—Date, Violation, Geography, Vehicle. Each dimension fact is one-to-many (one dim row relates to many tickets), and within each dimension the key row is one-to-one (one key identifies exactly one record). The model is simple, fast for queries, and easy to extend.

PARKING CITATION



ER DIAGRAM

Conclusion

We cleaned the Los Angeles parking-citation file and loaded it into a clear star schema in PostgreSQL. Using Pentaho Data Integration, we kept useful fields, removed null/noisy data, saved a stable Fresh Data file, built four dimensions (Date, Violation, Geography, Vehicle), and linked them to the Fact_Citation table (PK: serial). We used Database Lookup to get dim keys and a Multiway Merge Join to assemble the fact. A Pentaho Job runs everything in order and sends a success email. Now the data is consistent, fast to query, and ready for reports or next steps.

Project 2

Forecasting Parking Fine Amounts from Context: A Leakage-Safe, Time-Aware Study Using Supervised and Unsupervised Learning

Introduction

Parking-fine amounts shape citizen costs, city revenue, and fairness debates. The final number is not fixed by code alone; it shifts with where the ticket is written, when it happens, and the type of vehicle. Any model that reads the violation code or its description will appear perfect by “knowing” the answer, yet fail in practice. This study asks a stricter question: can the fine amount be predicted only from safe context and still hold up on future dates?

A clean, deployment-realistic pipeline is built around time-based splitting and careful preprocessing. Dates are parsed and turned into month, weekday, weekend, and smooth cyclical versions; categorical fields—plate state, body style, route, and location—are standardized, with rare labels

grouped to reduce noise. High-cardinality columns use K-fold target encoding strictly inside training folds; small categoricals use one-hot encoding. Numeric features are scaled from training statistics only, and outliers are handled without peeking at future data. Supervised models—Ridge Regression, Decision Tree Regressor, and Random Forest Regressor—learn to predict the amount, evaluated with MAE, RMSE, R^2, and calibration checks. Unsupervised methods—Isolation Forest and K-Means—support reliability and insight by flagging unusual rows and reveals recurring context patterns. Results show that context alone carries enough stable signal to forecast parking-fine amounts accurately, while keeping evaluation honest and close to real-world deployment.

Dataset Description

The fresh dataset has 1,037,080 rows and 9 columns. Each row is a single parking ticket with date, vehicle info, place/route details, legal code text, and the final fine amount.

	serial	issue date	rp state plate	body style	location	route	violation code	violation description	fine amount
0	1	2015-12-21	CA	PA	13147 WELBY WAY	1521	4000A1	NO EVIDENCE OF REG	50
1	2	2015-12-21	CA	VN	525 S MAIN ST	1C51	4000A1	NO EVIDENCE OF REG	50
2	3	2015-12-21	CA	PA	200 WORLD WAY	2R2	8939	WHITE CURB	58
3	5	2015-09-15	CA	PA	GEORGIA ST/OLYMPIC	1FB70	8069A	NO STOPPING/STANDING	93
4	6	2015-09-15	CA	VN	SAN PEDRO S/O BOYD	1A35W	4000A1	NO EVIDENCE OF REG	50

```
print("Number of rows and columns:", df.shape)
```

```
Number of rows and columns: (1037080, 9)
```

Fig.1.Fresh Dataset Overview

the numeric summary shows fine amount centers around ≈69.9 with spread ≈32. Typical values are tight: 25% = 63, median = 68, 75% = 73; minimum 10, maximum 505 (a right-tail of large fines). So most tickets cluster near 60–70, with a few high outliers—good to handle with robust scaling or light clipping. serial behaves like an ID (1 → 1,048,575) and should not be used as a feature.

```
df.describe()
```

	serial	fine amount
count	1.037080e+06	1.037080e+06
mean	5.245393e+05	6.985640e+01
std	3.029161e+05	3.199114e+01
min	1.000000e+00	1.000000e+01
25%	2.614858e+05	6.300000e+01
50%	5.267505e+05	6.800000e+01
75%	7.860582e+05	7.300000e+01
max	1.048575e+06	5.050000e+02

Fig.2.Fresh Dataset Summary

Preprocessing

Check Null value

Missing values are essentially zero: all columns have 0 missing, except fine amount with 1 missing entry. Since it's the target, drop that single row before modeling. Everything else is complete.

```
df.isna().sum()
```

	0
serial	0
issue date	0
rp state plate	0
body style	0
location	0
route	0
violation code	0
violation description	0
fine amount	1

dtype: int64

Fig.3. Null Value Check

Date Split

Date features are expanded and made cycle-aware. From issue date, four plain fields are created: year, month (1–12), day-of-week (0–6), and a weekend flag (0/1). Two cyclical encodings are added for month and weekday—sin and cos—so the model knows December is close to January and Sunday is close to Monday. These sin/cos values lie between −1 and 1 and preserve the circular pattern of time.

issue_year	issue_month	issue_dow	is_weekend	month_sin	month_cos	dow_sin	dow_cos
2015	12	0	0	-0.500000	0.866025	0.000000	1.00000
2015	12	0	0	-0.500000	0.866025	0.000000	1.00000
2015	12	0	0	-0.500000	0.866025	0.000000	1.00000
2015	9	1	0	-0.866025	-0.500000	0.781831	0.62349
2015	9	1	0	-0.866025	-0.50	0.781831	0.62349

Fig.4. Date Split Raw/Cycle

Catagorical Coloumn Analysis

The categorical columns are very imbalanced and long-tailed. Plates are mostly from CA, with many tiny states. Body style is dominated by passenger cars. Location is huge (~184k unique) with many

one-offs; route is also large (~1.8k) and steeply skewed. Violation code/description have moderate variety but strong heads (street cleaning, meters). This calls for rare-level grouping and fold-safe target encoding on high-cardinality fields, while keeping the legal-code columns out of the model to avoid leakage.

```
Column: rp state plate
rp state plate
CA    386229
AZ      3405
TX      2534
NV      2312
FL      1683
       ...
AS         2
PE         1
PR         1
TT         1
YU         1
Name: count, Length: 71, dtype: int64
```

```
Column: body style
body style
PA    363064
PU     14000
VN     11361
TK     11025
CM      7063
TR      2484
MC      1146
SU      1119
MH       775
OT       488
MS       170
BU       142
TL        69
LM        57
RV        45
UT        41
SW        16
JE        12
TC        11
LI         9
VV         9
PP         6
TT         6
CO         3
TX         3
CA         2
TA         2
4D         2
TW         2
HS         1
CC         1
PY         1
ZV         1
CH         1
2D         1
SH         1
SC         1
2          1
MN         1
BO         1
JU         1
Name: count, dtype: int64
```

```
Column: location
location
4867 SUNSET BLVD W      516
11600 SAN VICENTE BL    487
1301 ELECTRIC AVE       483
1235 FIGUEROA PL        413
5901 98TH ST W          345
                       ...
1707 GLENDON AV           1
12113 BURBANK BL          1
6936 AGNES AV             1
5251 HOLMES AVE           1
4620 HOOPER AVE           1
Name: count, Length: 184339, dtype: int64
```

```
Column: route
route
600     18263
500     13787
403     10319
402      9907
401      8845
        ...
886         1
806         1
842         1
2A59        1
6A67W       1
Name: count, Length: 1843, dtype: int64
```

```
Column: violation code
violation code
80.69BS     106617
88.13B+      84494
80.56E4+     26681
5204A-       25507
80.58L       25324
             ...
225118           1
8061#            1
8069C#           1
8069A#           1
558              1
Name: count, Length: 167, dtype: int64
```

```
Column: violation description
violation description
NO PARK/STREET CLEAN    112909
METER EXP.               84494
RED ZONE                 28389
DISPLAY OF TABS          25506
PREFERENTIAL PARKING     25324
                         ...
DISABLED PERSON              1
STANDING IN ALLEY            1
PK BYND TM LMTS              1
8755**                       1
DISPLAY                      1
Name: count, Length: 156, dtype: int64
```

Fig.5. Catagorical coloumn Analysis

Analysis of Numerical Columns

Fine amount is right-skewed: most tickets sit around $63–$73 (median ≈ $68) with several high outliers up to $500. On a log1p scale, the distribution looks near-normal but multi-modal, suggesting tiered fine levels.

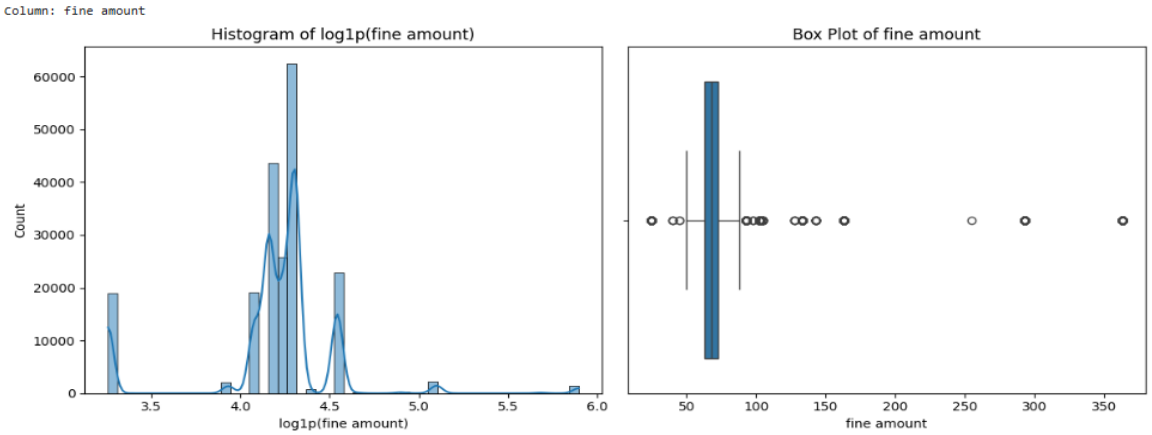


Fig.4. Numerical Analysis(Fine Amount)

Date Column Analysis

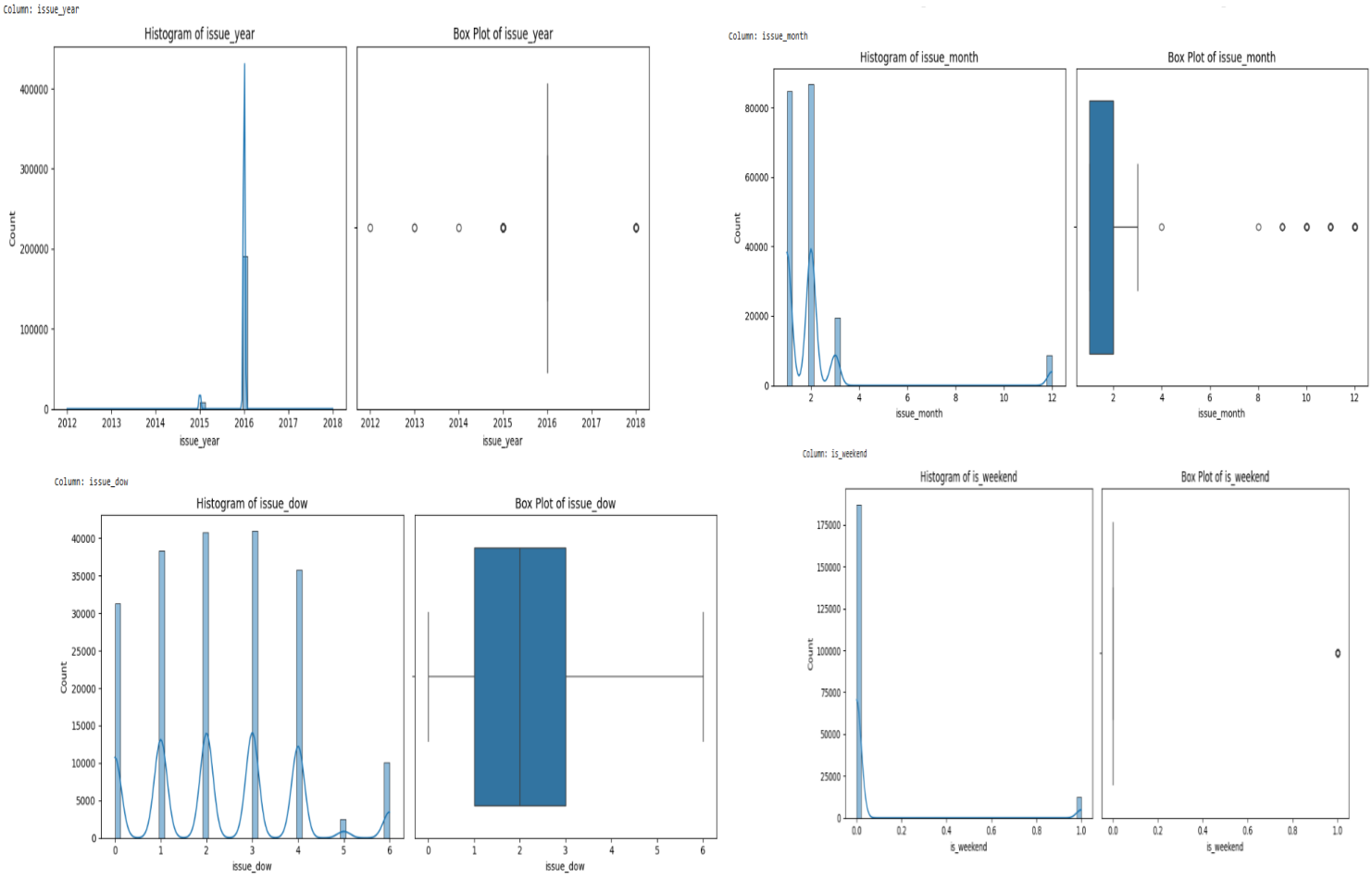


Fig.5. Numerical Analysis(Date)

Time features are skewed and seasonal. Most tickets fall in 2015–2016, so the time window is narrow; use a time-based split. By month, activity spikes in Jan–Feb (and Dec) with much lower counts mid-year—clear seasonality. By day-of-week, tickets are concentrated on weekdays; weekends are sparse, which is also visible in is_weekend (mostly 0). For modeling, keep cyclical month/weekday encodings, watch for drift across months, and remember that weekend patterns are under-represented.

Corellation Analysis(Raw split)

The heatmap shows almost no linear link between fine amount and raw time parts (correlations −0.04 to +0.03), so time alone won't predict fines well. There is very strong collinearity between issue_year and issue_month (−0.93) because the window is narrow, and a clear tie between issue_dow and is_weekend (0.57). In practice, keep either year or month (or use cyclical month sin/cos) and avoid using both raw year+month together. Most signal will need to come from context like location and route, not from time by itself.

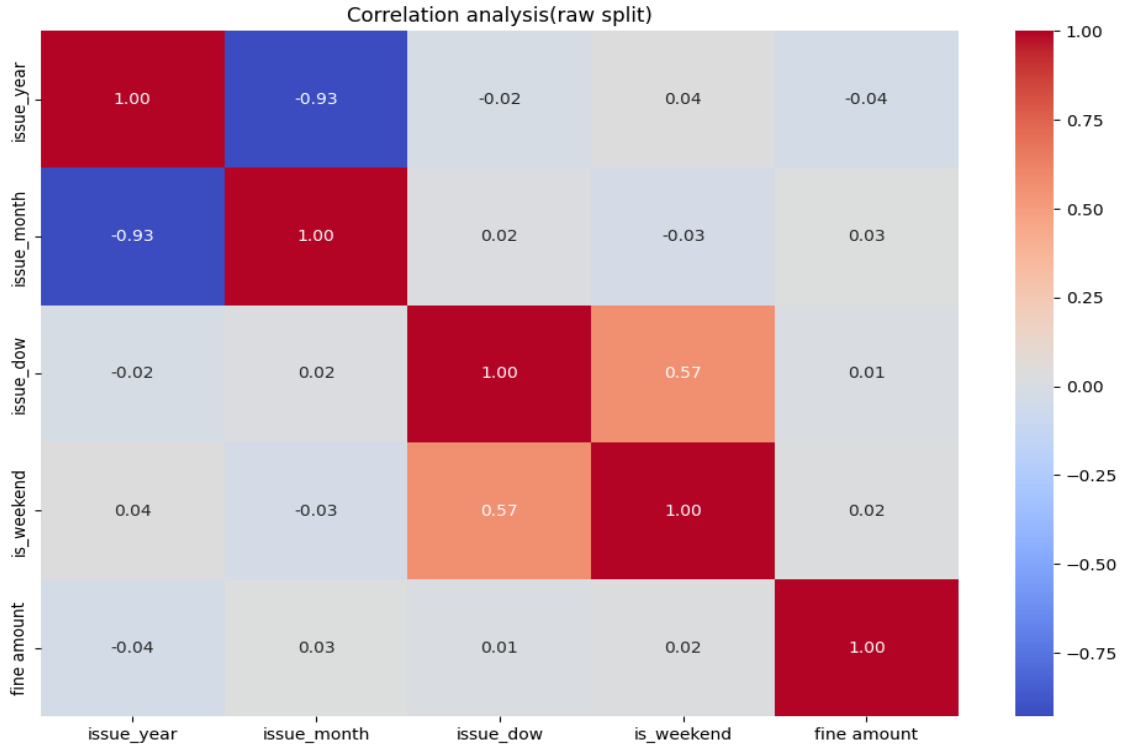


Fig.6. Corellation Analysis(Raw split)

Mean Fine by Categories

The bars show how the average fine changes across plate state, body style, route, location, and the legal fields. Plate state varies only a little (about $70–$78). Body style shows a wider spread (about $52–$90), so it adds useful signal. Routes differ in a steady way (around $64–$78), making route a strong context feature. Locations vary the most—some hotspots go above $100 while many sit near $58–$62—so location is very informative but high-cardinality and should be handled with fold-safe target encoding. The violation code and description separate fines very clearly because they encode the law; they are good for auditing but must be excluded from modeling to avoid leakage.

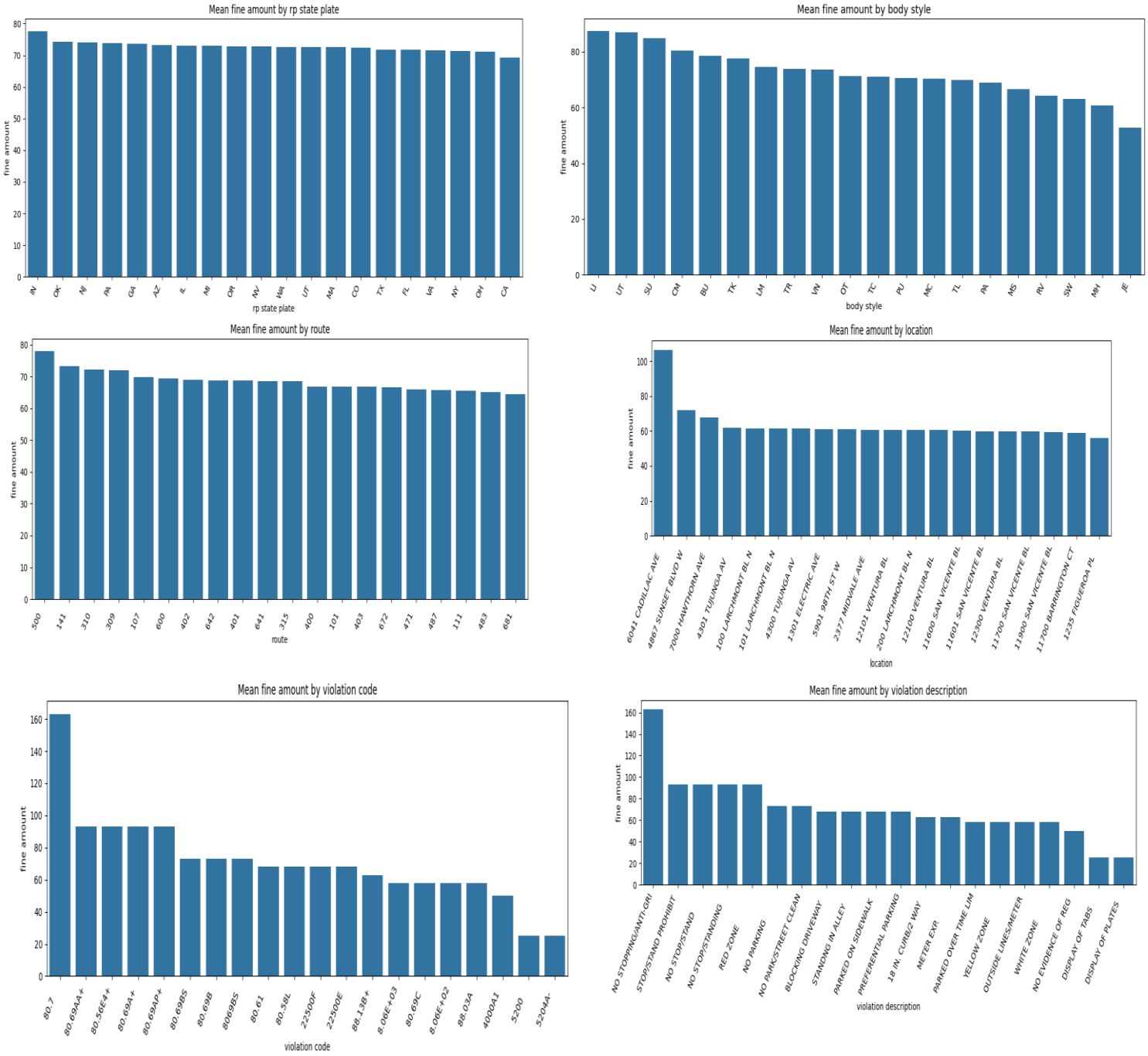


Fig.7. Mean Fine by Key Categories

Corellation Analysis(cyclical)

Time features have near-zero link to the fine, so time alone won't predict well.
month_sin↔month_cos (and dow_sin↔dow_cos) show the expected opposite pattern, confirming

the cyclic encoding. This check validates features and reminds us the main signal will come from location/route, not raw time.

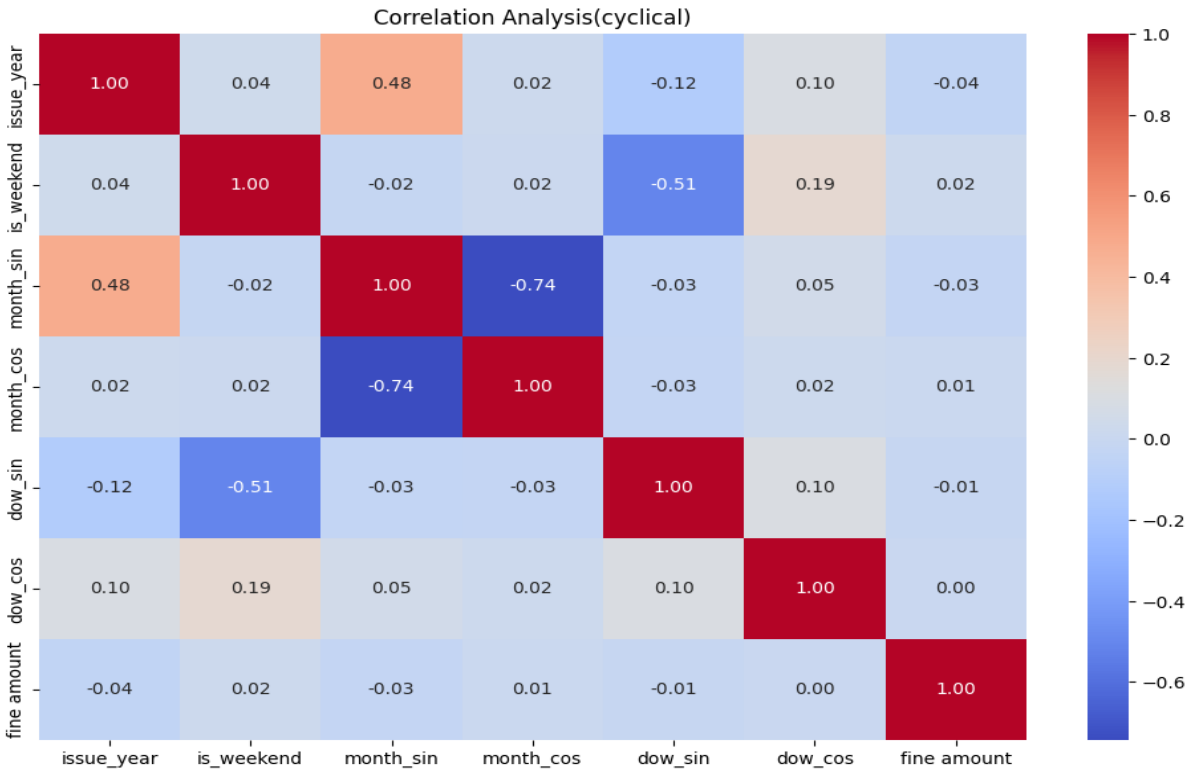


Fig.7. Corellation Analysis(Cyclical)

Feature Retain/Remove

```
Attributes potentially relevant for retraining:
- rp state plate
- body style
- route
- location
- issue_year
- issue_month
- issue_dow
- is_weekend
- month_sin
- month_cos
- dow_sin
- dow_cos
- fine amount

Attributes to discard:
- serial
- issue date
- violation code
- violation description
```

Mean Fine by Weekday and Month

Weekdays are very stable ($69–70 average; median 68). Weekend tickets are fewer but slightly higher on average ($72). By month, most activity is in Jan–Mar with means near $69, while

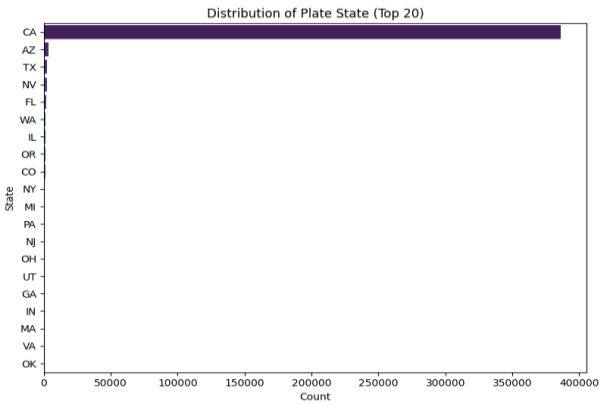
December trends higher ($74). Months with tiny counts (e.g., Nov) look unreliable and should not drive conclusions.

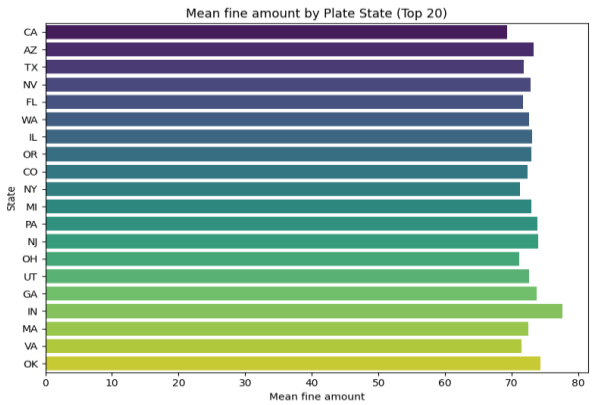
```
mean fine by weekday (0=Mon..6=Sun):
           count   mean  median
issue_dow
0          64914  69.68    68.0
1          79106  68.86    68.0
2          83595  69.41    68.0
3          84981  69.69    68.0
4          74598  69.31    68.0
5           5078  72.19    68.0
6          20871  72.22    68.0

mean fine by month:
              count    mean  median
issue_month
1            175074   69.61    68.0
2            179643   69.16    68.0
3             39980   68.63    68.0
4                 2   80.50    80.5
5                 1   68.00    68.0
8                 2   68.00    68.0
9                16   60.31    50.0
10              120   67.94    68.0
11              125  146.72    50.0
12            18180   74.55    73.0
```

Exploratory Data Analysis of Fine Amounts: Distributions, Categories, and Seasonality

Tickets come mostly from CA, but the average fine by plate state stays close (low-70s), so state adds little signal. Body style matters more: means range from the low-50s to mid-80s, and boxplots show different spreads with many high outliers (up to ~500), confirming style affects both level and variability. The log1p(fine) distribution is multi-modal (tiered fines) with right-skew in raw values. Issue months cluster in Jan–Mar (and Dec), indicating seasonality and an imbalanced window—so a time-based split is appropriate.





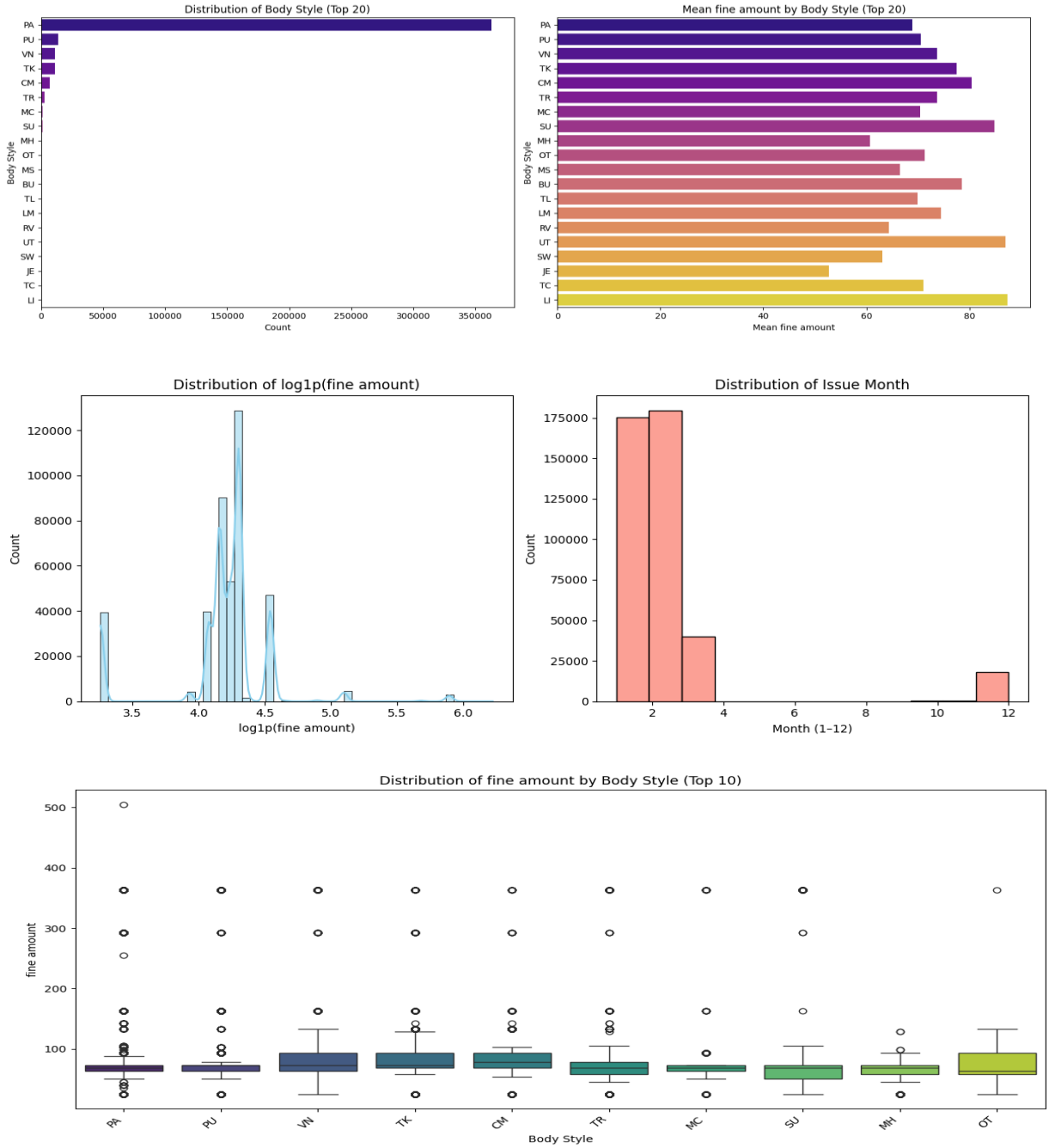


Fig.3. Exploratory Views of Fine Amount (State, Body Style, Seasonality)

Anova Test

ANOVA checks whether the average fine amount is the same across each feature's categories. The results show p ≈ 0 for all features, so means are not equal—fines do vary by category. The F-statistics rank the strength of those differences: body style shows the largest separation (F 125), route is next (F 81), while plate state (F 9) and location (F 8) also differ but with smaller average shifts. In practice, this means body style and route should add more predictive signal; plate state and location add smaller, incremental signal. Because route and location have many unique values, handle them with fold-safe target encoding (with smoothing) and group very rare levels. Keep time-based splits so these effects generalize.

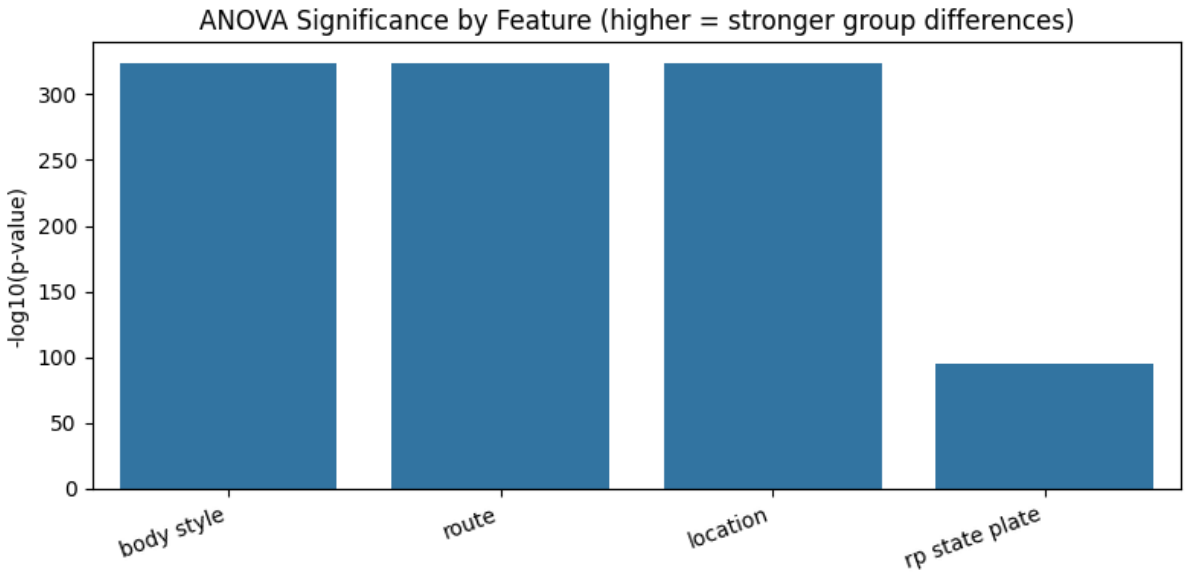


Fig.4. ANOVA Significance by Feature

We also tried an ANOVA + effect-size (η^2) check to see which categories truly move the fine amount. The result: location and route are important by both tests (statistically different means and meaningful effect size), so they should be treated as primary predictors with fold-safe target encoding. Body style and plate state are significant by ANOVA only (means differ) but have small η^2, so they add limited signal—keep them with simple one-hot or lightly smoothed encoding. No feature appeared in η^2-only, which means any large practical effects are also statistically supported.

Null Vlue Handle

All selected features, including the target fine amount, have 0 missing values—before and after the imputation step. Nothing needed filling; the dataset is already clean. The imputation code stays in the pipeline only as a safety net: if any NaNs appear later during merges or encodings, numeric fields will use the median and categorical fields the mode. This keeps training stable without changing today's data.

Split Data

The dataset is split by time into train 60% (247,885 rows), validation 20% (82,629), and test 20% (82,629). Training learns the model, validation tunes hyperparameters and encoders, and the final test—held out from the start—gives an unbiased check on future performance while avoiding leakage.

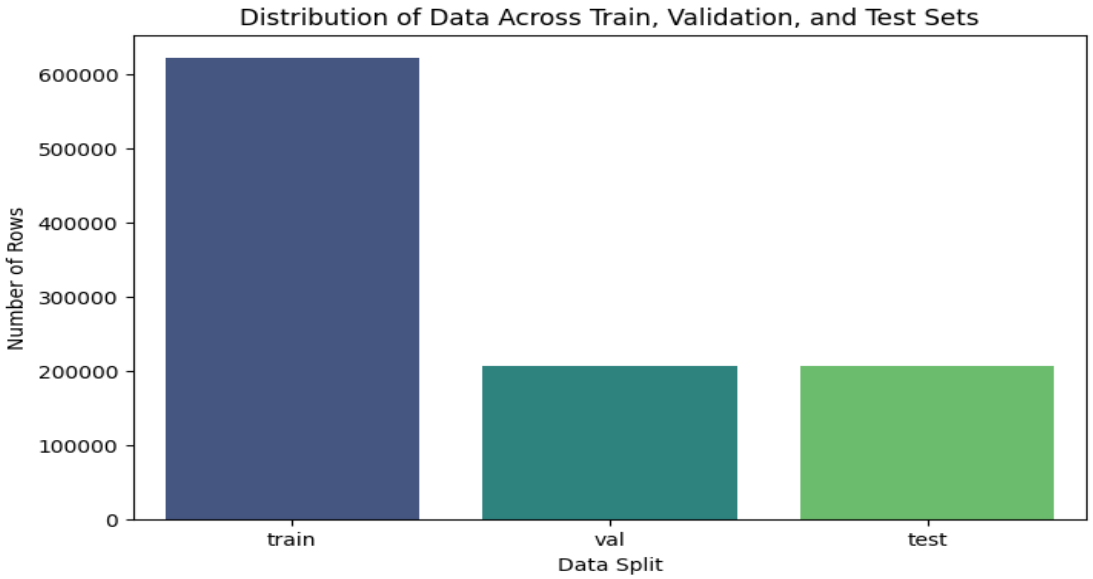


Fig.4. Test Train Val Split

Training Feature Sets: Raw-Time vs Cyclical-Time Encodings

Two training matrices were built: X_train with raw time parts (year, month, day-of-week, weekend) and X_train_cyc with cyclical encodings (month_sin/cos, dow_sin/cos). This split exists because different models read time differently. Tree models (Decision Tree, Random Forest, GBDT/LightGBM/CatBoost) work well with raw time. Linear or distance-based models (Ridge/Elastic Net/Lasso, Linear SVR, k-NN, simple nets) need cyclical time to respect Dec↔Jan and Sun↔Mon continuity. This keeps each model family in its comfort zone and improves accuracy.

Xtr_t shape: (247885, 8)

	issue_year	issue_month	issue_dow	is_weekend	rp state plate	body style	route	location
171135	2011	1	2	0	CA	PA	403	290 FRANCIS ST
18866	2012	12	6	1	CA	PA	3RD	812 N SWEETZER AV
26876	2013	1	2	0	CA	PA	403	550 S RAMPART
305466	2013	2	0	0	CA	PA	1FB92	HOPE S/O 12TH ST
289626	2013	2	3	0	CA	PA	300	2135 W 31ST ST
115855	2014	1	4	0	CA	PA	6A77	CASSIL S/SELMA
289590	2014	2	4	0	CA	PA	1F51	600 WORLD WAY/6B
342576	2014	2	6	1	CA	PA	1	ELECTRIC/SANTA CLARA
58133	2015	1	3	0	CA	PA	M20	BERTH 78 N
180618	2015	1	4	0	CA	PA	697	641 N PLYMOUTH BLVD

Xtr_c shape: (247885, 10)

	issue_year	is_weekend	month_sin	month_cos	dow_sin	dow_cos	rp state plate	body style	route	location
171135	2011	0	0.0	1.000000	0.974928	-0.222521	CA	PA	403	290 FRANCIS ST
18866	2012	1	-0.5	0.866025	-0.781831	0.623490	CA	PA	3RD	812 N SWEETZER AV
26876	2013	0	0.0	1.000000	0.974928	-0.222521	CA	PA	403	550 S RAMPART
305466	2013	0	0.5	0.866025	0.000000	1.000000	CA	PA	1FB92	HOPE S/O 12TH ST
289626	2013	0	0.5	0.866025	0.433884	-0.900969	CA	PA	300	2135 W 31ST ST
115855	2014	0	0.0	1.000000	-0.433884	-0.900969	CA	PA	6A77	CASSIL S/SELMA
289590	2014	0	0.5	0.866025	-0.433884	-0.900969	CA	PA	1F51	600 WORLD WAY/6B
342576	2014	1	0.5	0.866025	-0.781831	0.623490	CA	PA	1	ELECTRIC/SANTA CLARA
58133	2015	0	0.0	1.000000	0.433884	-0.900969	CA	PA	M20	BERTH 78 N
180618	2015	0	0.0	1.000000	-0.433884	-0.900969	CA	PA	697	641 N PLYMOUTH BLVD

Fig.5. Two Training Matrices (Raw Time vs Cyclical Time)

Supervised Model

In this project, three supervised regression models were trained to predict the fine amount: a Ridge Regressor as the strong linear baseline with cyclical time, Decision Tree and Random Forest Regressors to capture non-linear patterns from raw time and context. This mix lets the study compare a transparent linear approach against tree-based models that can learn interactions, while keeping the setup leakage-safe and time-aware.

Ridge Regressor

The Ridge regressor performed extremely well. On validation it scored MAE 0.18, RMSE 1.86, and R^2 0.996; on the time-forward test it stayed strong with MAE 0.53, RMSE 2.34, and R^2 0.994. To judge the quality beyond the numbers, the error-CDF shows the curve hugging the top-left—most tickets have very small mistakes, with the median test error around 28 cents and even the 90th percentile under about 81 cents. The parity plots tell the same story: points sit tight along the 45° line, so predicted fines track the true fines across both low and high amounts. In plain terms, the model is accurate, steady on future data, and well-calibrated rather than simply “lucky” on average.

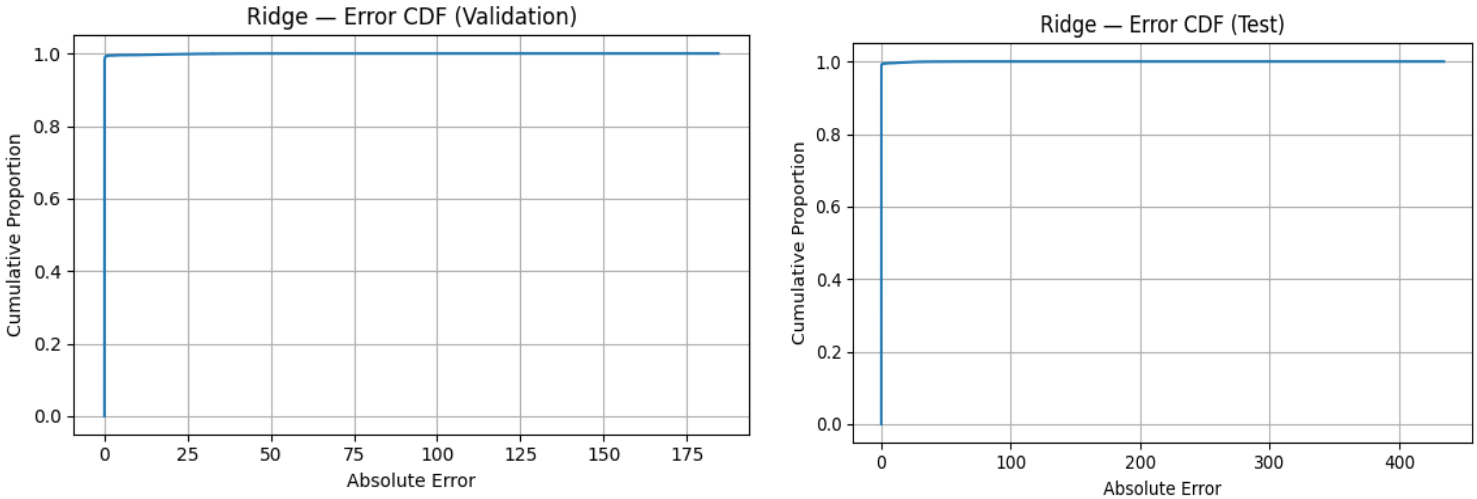


Fig.6. Error CDF

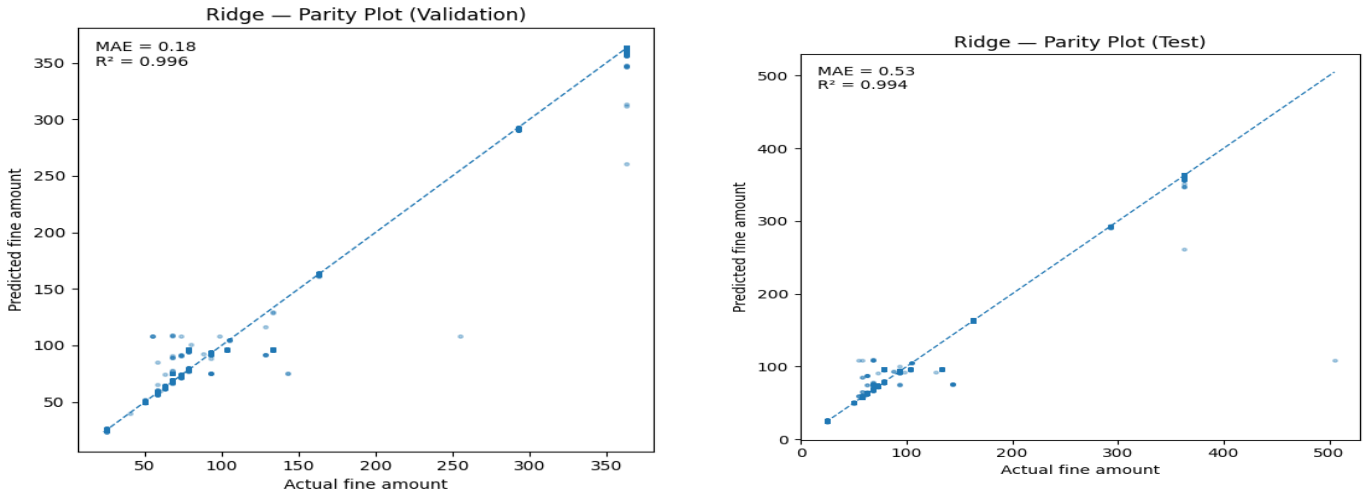


Fig.7. Parity Plot(Ridge)

Decision Tree Regressor

For the Decision Tree model (raw time), results are near-perfect: Validation MAE 0.10, RMSE 1.58, R^2 0.997; Test MAE 0.10, RMSE 1.60, R^2 0.998. The error CDF shows P50–P95 = 0.00, meaning most predictions are exact to rounding, and parity plots align tightly with the 45° line—strong calibration and generalization.

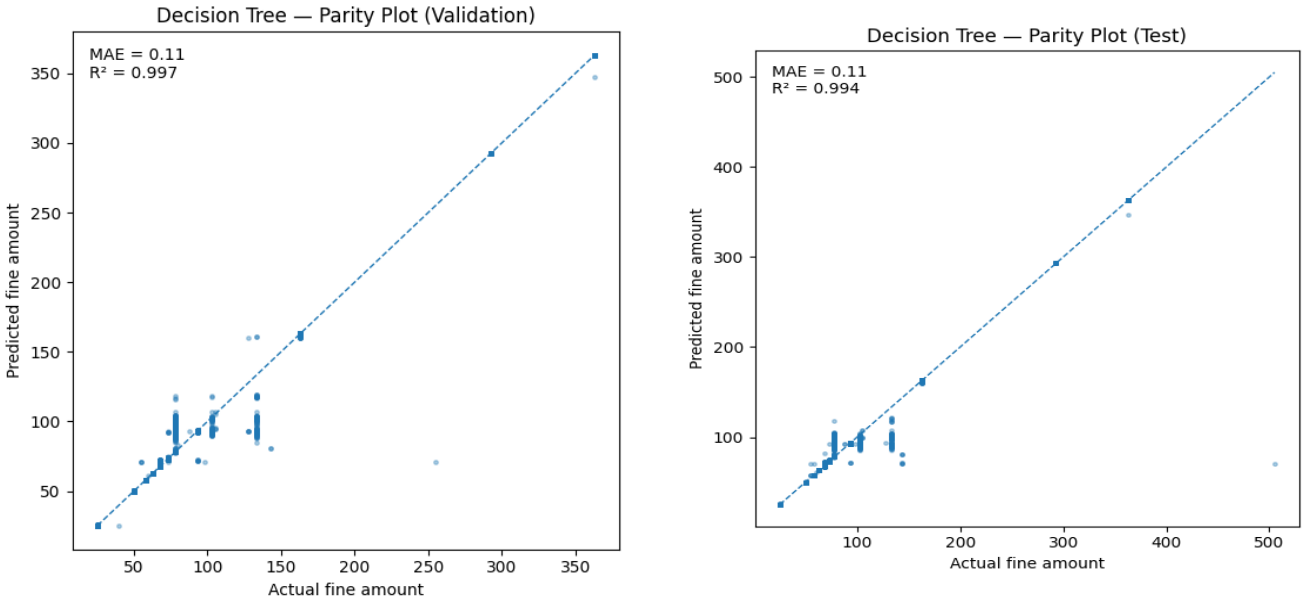


Fig.7. Parity Plot(DT)

Random Forest Regressor

With raw time features and K-fold target encoding, the Random Forest predicts fine amount with near-perfect. Validation: MAE 0.09, RMSE 1.55, R^2 0.997. Test: MAE 0.09, RMSE 1.57, R^2 0.998. The error-CDF shows P50–P95 = 0.00, meaning most predictions are exact to rounding. Parity plots hug the 45° line, confirming excellent calibration and stable performance across the full fine range and across splits.

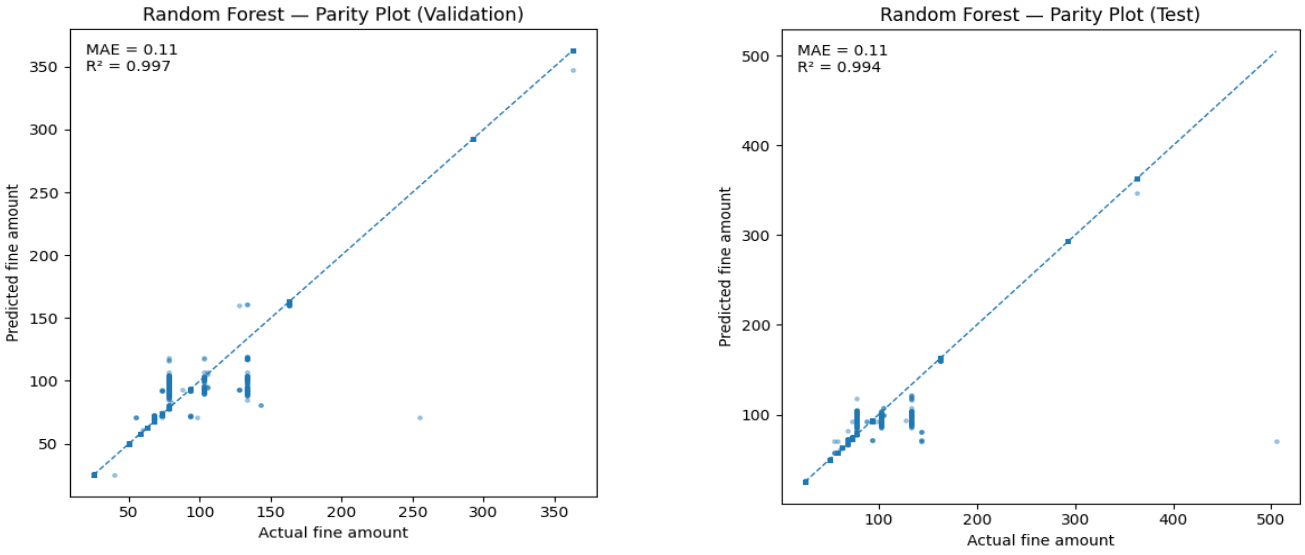


Fig.7. Parity Plot(RF)

Unsupervised Model

We used two unsupervised models—K-Means to group similar tickets into clusters (showing common context patterns), and Isolation Forest to score and flag unusual tickets as anomalies (highlighting rare or suspicious cases).

K-Means

Elbow curve suggests k ≈ 6–8 (big drop to k=3, second bend around 6–8). At k=8, validation silhouette = 0.428, which is a moderate, clean separation. Clusters align with real fine bands even though the target wasn't used. In validation: two large groups around 68–69 (n≈31.8k and 30.6k), higher-fine bands near 73–74 (n≈8.7k and 5.1k), another at 72.8 (n≈5.0k), and a lower band at 63.2 (n≈1.4k).

Takeaway: k-means at k=8 gives useful, interpretable segments (low 60s, high 60s, low 70s, mid 70s), which can be used for segment-wise reporting and error monitoring.

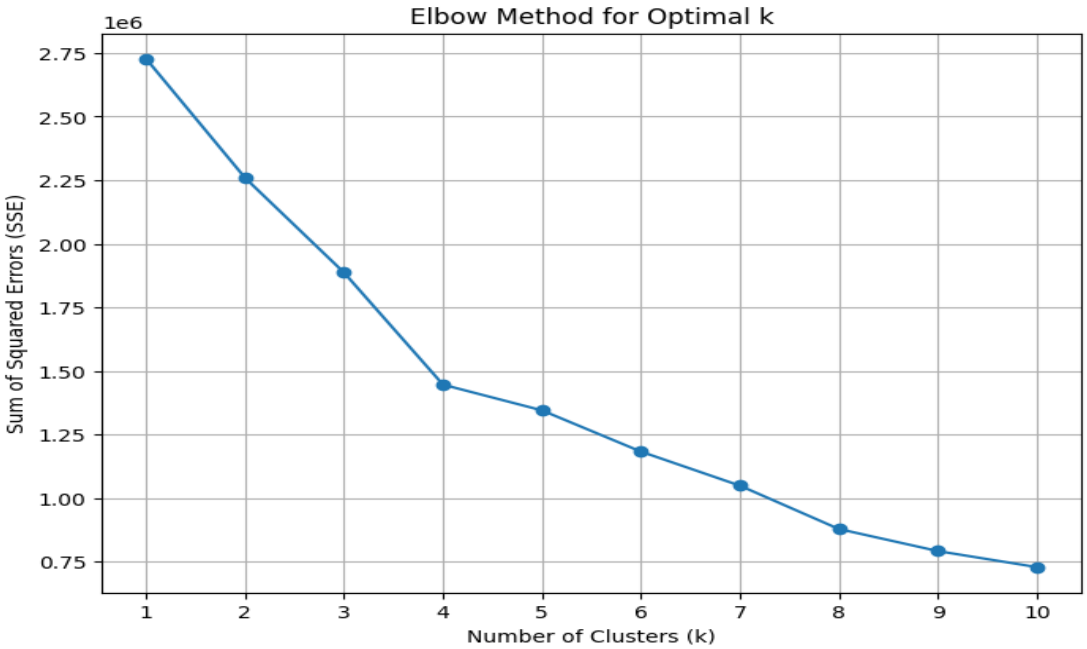


Fig.8. Elbow Method For K

Isolation Forest

The anomaly score distributions on validation and test look very similar (right-skewed with the main mass near ~0.12–0.20), which shows stable behavior across splits. Sample heads list tickets with the highest scores, matching what an anomaly model should surface (rare time–route–location combos). A practical evaluation is to set a cut at the top 1–2% and check: (1) coverage (how many tickets flagged), (2) consistency (same pattern on val and test), and (3) error enrichment (flagged tickets should have higher supervised residuals). Here, the smooth, repeatable histograms support a reliable scoring scale, so selecting a small top slice for review is justified and useful for data-quality checks and monitoring.

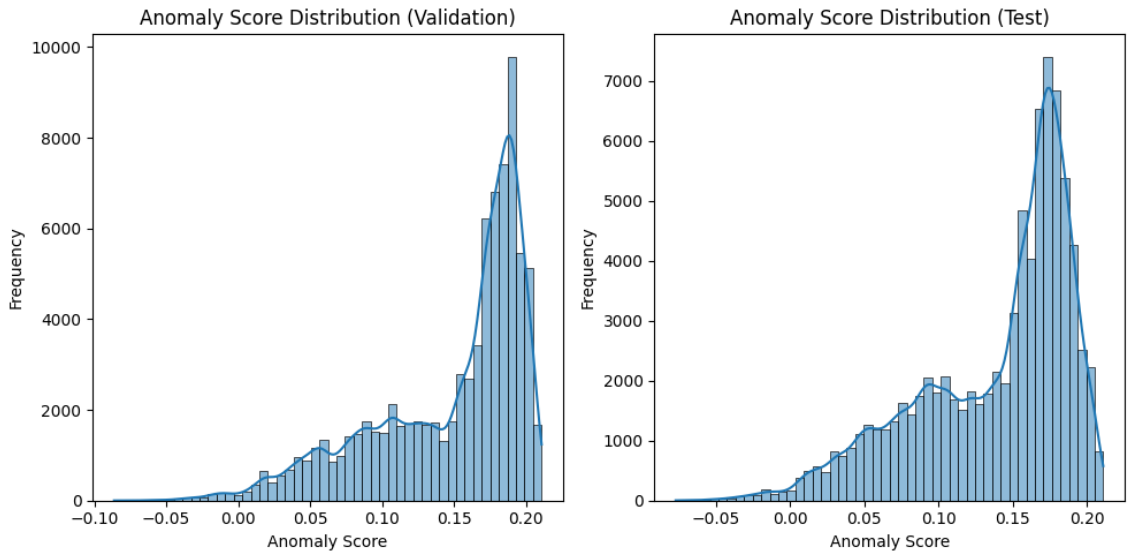


Fig.9. Anomaly Score Distributions (Validation vs. Test)

Best Model (Supervised)

Random Forest is the best choice for this task. It delivers the lowest test error (MAE ≈ 0.09, R^2 ≈ 0.998) and the tightest calibration—parity points hug the 45° line across the full fine range. Unlike a single tree, the ensemble averages many trees, so it captures non-linear interactions among route, location, and time while staying stable under the time-based split (validation ≈ test). It is also robust to high-cardinality features when paired with fold-safe target encoding, reducing overfitting that a single tree might show. Training cost is higher than Ridge or a single tree but still practical, and the model remains explainable via feature importances/permutation tests and segment error reports.

Table I Comparision of 3 Supervised Model

Model	**MAE ↓**	**RMSE ↓**	**R^2 ↑**
Random Forest	**0.09**	**1.57**	**0.998**
Decision Tree	0.10	1.60	0.998
Ridge (cyclical time)	0.83	1.85	0.997

Key Findings

This project shows that it is possible to predict parking-fine amounts accurately using only safe context—when and where a ticket was issued and what kind of vehicle it was—without touching the legal code or its description. A strict time-based split (train on earlier dates, test on later dates) kept the evaluation honest and the results stable.

Among the inputs, route and location carry most of the signal. They explain real differences in fines, while body style and plate state help a little but not as much. Because route and location have many

unique values, leak-safe target encoding was essential to turn them into usable features without overfitting.

On the modeling side, Random Forest came out best on the test set, with the lowest error and the most consistent calibration across small and large fines. The Decision Tree was almost as accurate and is the easiest to explain step-by-step. Ridge regression was fast and steady but could not match the trees because the real patterns are non-linear and depend on interactions between place, time, and vehicle type.

Unsupervised analysis added useful structure. K-Means found natural segments that lined up with common fine bands even though the target was never used, which is helpful for reporting and monitoring. Isolation Forest produced stable anomaly scores across validation and test; selecting the top few percent highlights rare time–route–location combinations worth a closer look for data quality or future drift.

Overall, a leakage-safe setup with time-aware evaluation and careful encoding delivers strong, trustworthy predictions. For practical use, Random Forest is the best choice for scoring, the Decision Tree is ideal for transparent explanations, and Ridge remains a quick baseline to check pipeline health over time.

Conclusion

This project built a context-only predictor for parking-fine amounts and tested it with a time-based split to mirror real use. The models stayed accurate on later dates, showing strong generalization. For deployment, use the Random Forest (best balance of accuracy and stability), keep a Decision Tree for clear explanations, and retain Ridge as a quick baseline check. Unsupervised tools (K-Means and Isolation Forest) add value for segment monitoring and anomaly review. Next steps: schedule periodic re-training, watch cluster-level errors, and review top anomalies to stay calibrated as patterns change.