Motor Vehicles Theft Data Analysis

In [2]:
```
from sqlalchemy import create_engine
import pandas as pd
```

In [3]:
```python
# Create database connection using SQLAlchemy
# engine = create_engine('mysql+pymysql://your_username:your_password@localhost/your_database')

engine = create_engine('mysql+pymysql://root:password@localhost/Mobile_theft')
```

1. locations Table

In [4]:
```python
query = """
SELECT * FROM locations LIMIT 10;
"""

df = pd.read_sql(query, engine)
df
```

Out[4]:

	location_id	region	country	population	density
0	101	Northland	New Zealand	201500	16.11
1	102	Auckland	New Zealand	1695200	343.09
2	103	Waikato	New Zealand	513800	21.50
3	104	Bay of Plenty	New Zealand	347700	28.80
4	105	Gisborne	New Zealand	52100	6.21
5	106	Hawke's Bay	New Zealand	182700	12.92
6	107	Taranaki	New Zealand	127300	17.55
7	108	ManawatÅ«-Whanganui	New Zealand	258200	11.62
8	109	Wellington	New Zealand	543500	67.52
9	110	Tasman	New Zealand	58700	6.10

2. make_details Table

In [5]:

```
query = """
SELECT * FROM make_details LIMIT 10;
"""

df = pd.read_sql(query, engine)
df
```

Out[5]:

	make_id	make_name	make_type
0	501	Aakron Xpress	Standard
1	502	ADLY	Standard
2	503	Alpha	Standard
3	504	Anglo	Standard
4	505	Aprilia	Standard
5	506	Atlas	Standard
6	507	Audi	Standard
7	508	Bailey	Standard
8	509	Bedford	Standard
9	510	Benelli	Standard

3. stolen_vehicles Table

In [19]:
```
query = """
SELECT * FROM stolen_vehicles LIMIT 10;
"""

df = pd.read_sql(query, engine)
df
```

Out[19]:

	vehicle_id	vehicle_type	make_id	model_year	vehicle_desc	color	date_stolen	location_id
0	1	Trailer	623	2021	BST2021D	Silver	11/5/21	102
1	2	Boat Trailer	623	2021	OUTBACK BOATS FT470	Silver	12/13/21	105
2	3	Boat Trailer	623	2021	ASD JETSKI	Silver	2/13/22	102
3	4	Trailer	623	2021	MSC 7X4	Silver	11/13/21	106
4	5	Trailer	623	2018	D-MAX 8X5	Silver	1/10/22	102
5	6	Roadbike	636	2005	YZF-R6T	Black	12/31/21	102
6	7	Trailer	623	2021	CAAR TRANSPORTER	Silver	11/12/21	114
7	8	Boat Trailer	623	2001	BOAT	Silver	2/22/22	109
8	9	Trailer	514	2021	7X4-6" 1000KG"	Silver	2/25/22	115
9	10	Trailer	514	2020	8X4 TANDEM	Silver	1/3/22	114

1. Find the total number of stolen vehicles per region

In [7]:
```python
query = """
SELECT l.region, COUNT(s.vehicle_id) AS total_stolen_vehicles
FROM stolen_vehicles s
JOIN locations l ON s.location_id = l.location_id
GROUP BY l.region
ORDER BY total_stolen_vehicles DESC;
"""

df = pd.read_sql(query, engine)
df
```

Out[7]:

	region	total_stolen_vehicles
0	Auckland	1630
1	Canterbury	660
2	Bay of Plenty	445
3	Wellington	417
4	Waikato	369
5	Northland	234
6	Gisborne	175
7	Otago	139
8	ManawatÅ«-Whanganui	139
9	Taranaki	112
10	Hawke's Bay	100
11	Nelson	92
12	Southland	26

- **Conclusion** – Auckland has the highest number of stolen vehicles.

- **Insights & Actions** – High-theft regions need increased police patrolling and surveillance to reduce vehicle theft.

2. Retrieve the most common vehicle color that gets stolen.

In [9]:
```
query = """
SELECT color, COUNT(*) AS stolen_count
FROM stolen_vehicles
GROUP BY color
ORDER BY stolen_count DESC
LIMIT 1;
"""

df = pd.read_sql(query, engine)
df
```

Out[9]:

	color	stolen_count
0	Silver	1272

- **Conclusion** – Silver-colored vehicles are stolen the most.
- **Insights & Actions** –Owners of silver vehicles should take extra precautions, such as installing GPS trackers.

3. Get the top 5 most stolen vehicle makes.

In [11]:
```
query = """
SELECT m.make_name, COUNT(s.vehicle_id) AS total_stolen
```

```
FROM stolen_vehicles s
JOIN make_details m ON s.make_id = m.make_id
GROUP BY m.make_name
ORDER BY total_stolen DESC
LIMIT 5;
"""

df = pd.read_sql(query, engine)
df
```

Out[11]:

	make_name	total_stolen
0	Toyota	716
1	Trailer	543
2	Nissan	482
3	Mazda	433
4	Ford	312

- **Conclusion** – The most frequently stolen vehicle make is identified.
- **Insights & Actions** – Manufacturers should improve anti-theft mechanisms in highly targeted models.

4. Find the average model year of stolen vehicles per region.

In [12]:

```
query = """
SELECT l.region, ROUND(AVG(s.model_year), 2) AS avg_model_year
FROM stolen_vehicles s
JOIN locations l ON s.location_id = l.location_id
GROUP BY l.region;
"""
```

```
df = pd.read_sql(query, engine)
df
```

Out[12]:

	region	avg_model_year
0	Auckland	2007.14
1	Gisborne	2003.13
2	Hawke's Bay	2005.15
3	Canterbury	2003.83
4	Wellington	2005.51
5	Otago	2002.11
6	ManawatÅ«-Whanganui	2003.88
7	Northland	2004.44
8	Bay of Plenty	2004.01
9	Waikato	2004.98
10	Nelson	2002.21
11	Taranaki	2004.34
12	Southland	2001.58

- **Conclusion** – Stolen vehicles' average model year varies by region.
- **Insights & Actions** – Older vehicles might lack security features, making them easier targets.

5. Determine the average time difference (in days) between stolen vehicles in each region.

In [64]:

```
query = """
WITH TheftDifferences AS (
    SELECT
        l.region,
        DATEDIFF(LEAD(date_stolen) OVER (PARTITION BY l.region ORDER BY date_stolen), date_stolen) AS days_between_thefts
    FROM stolen_vehicles sv
    JOIN locations l ON sv.location_id = l.location_id
)
SELECT
    region,
    AVG(days_between_thefts) AS avg_days_between_thefts
FROM TheftDifferences
WHERE days_between_thefts IS NOT NULL
GROUP BY region;
"""

df = pd.read_sql(query, engine)
df
```

Out[64]:

	region	avg_days_between_thefts
0	Auckland	0.1111
1	Bay of Plenty	0.4077
2	Canterbury	0.2731
3	Gisborne	1.0230
4	Hawke's Bay	1.8182
5	ManawatÅ«-Whanganui	1.3043
6	Nelson	1.9670
7	Northland	0.7725
8	Otago	1.3043
9	Southland	6.6400
10	Taranaki	1.5676
11	Waikato	0.4918
12	Wellington	0.4327

- **Conclusion** – Regions with a lower average number of days between thefts experience frequent incidents, while higher averages indicate lower theft activity. Specific colors or models may be more targeted.
- **Insights & Actions** – High-theft regions need increased patrolling and real-time monitoring. Identify peak theft seasons to enhance security. Public awareness campaigns should educate vehicle owners on anti-theft measures. Insurance companies can adjust premiums based on theft frequency. Law enforcement can use this data for predictive policing and better resource allocation.

6. Find the top 3 most stolen vehicle types.

In [23]:
```
query = """
SELECT vehicle_type, COUNT(*) AS stolen_count
FROM stolen_vehicles
GROUP BY vehicle_type
ORDER BY stolen_count DESC
LIMIT 3;
"""

df = pd.read_sql(query, engine)
df
```

Out[23]:

	vehicle_type	stolen_count
0	Stationwagon	945
1	Saloon	851
2	Hatchback	644

- **Conclusion** – Certain vehicle types are targeted more frequently.
- **Insights & Actions** – Owners of these vehicle types should invest in advanced security systems.

7. Rank stolen vehicle makes by frequency using window functions.

In [33]:
```
query = """
SELECT make_name, stolen_count,
       RANK() OVER (ORDER BY stolen_count DESC) AS rank_position
```

```
FROM (
    SELECT m.make_name, COUNT(s.vehicle_id) AS stolen_count
    FROM stolen_vehicles s
    JOIN make_details m ON s.make_id = m.make_id
    GROUP BY m.make_name
) ranked;
"""

df = pd.read_sql(query, engine)
df
```

Out[33]:

	make_name	stolen_count	rank_position
0	Toyota	716	1
1	Trailer	543	2
2	Nissan	482	3
3	Mazda	433	4
4	Ford	312	5
...	...	...	...
133	Toko	1	82
134	Nissan Diesel	1	82
135	Alpha	1	82
136	Niu	1	82
137	Caterpillar	1	82

138 rows × 3 columns

- **Conclusion** – Ranking helps prioritize high-risk vehicle brands.

- **Insights & Actions** – Law enforcement can focus efforts on preventing theft of top-ranked brands.

8. Find the first and last reported stolen vehicle.

In [ ]:
```
#First reported stolen vehicle
query = """
SELECT vehicle_id, date_stolen
FROM stolen_vehicles
ORDER BY date_stolen ASC
LIMIT 1;
"""

df = pd.read_sql(query, engine)
df
```

In [36]:
```
#Last reported stolen vehicle
query = """
SELECT vehicle_id, date_stolen
FROM stolen_vehicles
ORDER BY date_stolen DESC
LIMIT 1;
"""

df = pd.read_sql(query, engine)
df
```

Out[36]:

	vehicle_id	date_stolen
0	1187	2022-04-06

- **Conclusion** – Helps analyze historical theft trends.

- **Insights & Actions** – Knowing when thefts started and peaked can help adjust security measures.

9. Use CASE to classify stolen vehicles by model year category.

In [37]:

```
query = """
SELECT vehicle_id, model_year,
       CASE
           WHEN model_year >= 2020 THEN 'New Model'
           WHEN model_year BETWEEN 2010 AND 2019 THEN 'Moderate Age'
           ELSE 'Old Model'
       END AS vehicle_category
FROM stolen_vehicles;
"""

df = pd.read_sql(query, engine)
df
```

Out[37]:

	vehicle_id	model_year	vehicle_category
0	1	2021	New Model
1	2	2021	New Model
2	3	2021	New Model
3	4	2021	New Model
4	5	2018	Moderate Age
...	...	...	...
4533	4534	2007	Old Model
4534	4535	2005	Old Model
4535	4536	2012	Moderate Age
4536	4537	2010	Moderate Age
4537	4538	2019	Moderate Age

4538 rows × 3 columns

- **Conclusion** – Segments stolen vehicles based on age.
- **Insights & Actions** – Identifies which age group of vehicles is most vulnerable to theft.

10. Identify vehicle makes that have been stolen in more than 2 regions.

In [39]:

```
query = """
SELECT m.make_name, COUNT(DISTINCT l.region) AS regions_affected
FROM stolen_vehicles s
```

```
JOIN make_details m ON s.make_id = m.make_id
JOIN locations l ON s.location_id = l.location_id
GROUP BY m.make_name
HAVING regions_affected > 2;
"""

df = pd.read_sql(query, engine)
df
```

Out[39]:

	make_name	regions_affected
0	Aprilia	6
1	Audi	8
2	Benelli	3
3	BMW	12
4	Briford	8
5	Caravan	6
6	Chevrolet	4
7	Chrysler	5
8	Daihatsu	9
9	Dodge	3
10	Factory Built	10
11	Ford	13
12	Forza	3
13	FOTON	4
14	Harley Davidson	5
15	Holden	13
16	Homebuilt	12
17	Honda	13
18	Hyosung	6
19	Hyundai	6
20	Isuzu	9
21	Kawasaki	5

	make_name	regions_affected
22	Kea	8
23	Keeway	4
24	Kia	3
25	KTM	6
26	Land Rover	3
27	Lexus	3
28	Mazda	13
29	Mercedes-Benz	6
30	Mini	3
31	Mitsubishi	13
32	Moped	5
33	Nissan	13
34	Peugeot	8
35	PGO	4
36	Piaggio	3
37	Pinto	5
38	Reid	4
39	Royal Enfield	3
40	Ssangyong	3
41	Subaru	13
42	Suzuki	13
43	Titan	5

	make_name	regions_affected
44	TNT Motor	7
45	Toyota	13
46	Trailer	13
47	Triumph	6
48	Vespa	3
49	Volkswagen	9
50	Yamaha	11

- **Conclusion** – Some vehicle brands have widespread theft occurrences.
- **Insights & Actions** – Insurance companies can adjust policy rates based on theft risk per make.

11. Find regions where theft density is highest compared to population.

In [41]:
```
query = """
SELECT l.region, COUNT(s.vehicle_id) / CAST(l.population AS FLOAT) AS theft_density
FROM stolen_vehicles s
JOIN locations l ON s.location_id = l.location_id
GROUP BY l.region, l.population
ORDER BY theft_density DESC;
"""

df = pd.read_sql(query, engine)
df
```

Out[41]:

	region	theft_density
0	Gisborne	0.003359
1	Nelson	0.001688
2	Bay of Plenty	0.001280
3	Northland	0.001161
4	Canterbury	0.001008
5	Auckland	0.000962
6	Taranaki	0.000880
7	Wellington	0.000767
8	Waikato	0.000718
9	Otago	0.000565
10	Hawke's Bay	0.000547
11	ManawatÅ«-Whanganui	0.000538
12	Southland	0.000254

- **Conclusion** – Some regions have higher theft density relative to population.
- **Insights & Actions** – High-density areas need better vehicle security awareness campaigns.

12. Create a CTE to find vehicles stolen more than once.

In [46]:
```
query = """
WITH multiple_stolen AS (
```

```
    SELECT vehicle_desc, COUNT(*) AS theft_count
    FROM stolen_vehicles
    GROUP BY vehicle_desc
    HAVING COUNT(*) > 1
)
SELECT * FROM multiple_stolen;
"""

df = pd.read_sql(query, engine)
df
```

Out[46]:

	vehicle_desc	theft_count
0	BST2021D	2
1	ASD JETSKI	3
2	YZF-R6T	2
3	BOAT	15
4	8X4 TANDEM	8
...	...	...
419	HI-LUX 2.4 C/C	2
420	JIMNY	2
421	COROLLA 1.6P GL HBAC	2
422	TRADER	2
423		18

424 rows × 2 columns

- **Conclusion** – Some vehicles have been stolen multiple times.

- **Insights & Actions** – Owners should take extra security measures, such as immobilizers.

13. Find the latest stolen vehicle for each vehicle make.

In [47]:
```
query = """
SELECT vehicle_desc, make_name, date_stolen
FROM (
    SELECT s.vehicle_desc, m.make_name, s.date_stolen,
           RANK() OVER (PARTITION BY m.make_name ORDER BY s.date_stolen DESC) AS rnk
    FROM stolen_vehicles s
    JOIN make_details m ON s.make_id = m.make_id
) ranked
WHERE rnk = 1;
"""

df = pd.read_sql(query, engine)
df
```

Out[47]:

	vehicle_desc	make_name	date_stolen
0	HUMBAUR	Aakron Xpress	2022-02-05
1	GTA-50	ADLY	2022-02-14
2		Alpha	2021-12-02
3	CARAVAN	Anglo	2021-12-15
4	PEGASO	Aprilia	2022-04-05
...	...	...	...
164	S80	Volvo	2022-03-14
165	JII	Voyager	2022-01-05
166	WR	Yamaha	2022-04-03
167		Zephyr	2022-02-25
168	ZN50QT-51A	Znen	2022-02-25

169 rows × 3 columns

- **Conclusion** – Identifies recent theft trends by make.
- **Insights & Actions** – Helps authorities track new theft patterns quickly.

14. Use a subquery to find stolen vehicles in the most affected region.

In [50]:

```
query = """
SELECT *
FROM stolen_vehicles
```

```
WHERE location_id = (
    SELECT location_id
    FROM stolen_vehicles
    GROUP BY location_id
    ORDER BY COUNT(vehicle_id) DESC
    LIMIT 1
);
"""

df = pd.read_sql(query, engine)
df
```

Out[50]:

	vehicle_id	vehicle_type	make_id	model_year	vehicle_desc	color	date_stolen	location_id
0	1	Trailer	623	2021	BST2021D	Silver	2021-11-05	102
1	3	Boat Trailer	623	2021	ASD JETSKI	Silver	2022-02-13	102
2	5	Trailer	623	2018	D-MAX 8X5	Silver	2022-01-10	102
3	6	Roadbike	636	2005	YZF-R6T	Black	2021-12-31	102
4	12	Trailer	538	2018	BRENT SMITH TRAILERS	Silver	2022-02-28	102
...	...	...	...	...	...	...	...	...
1625	4523	Trailer	549	1993		Silver	2021-12-12	102
1626	4529		507	2011	A8	Grey	2021-12-14	102
1627	4530		512	2009	335i	Black	2022-02-22	102
1628	4532		589	2021	NQI	White	2022-03-07	102
1629	4535		520	2005	30600	Yellow	2021-11-23	102

1630 rows × 8 columns

- **Conclusion** – Identifies the region with the most thefts.

- **Insights & Actions** – Extra security should be deployed in high-theft regions.

15. Calculate the percentage contribution of each vehicle make to total thefts.

In [52]:
```
query = """
SELECT m.make_name,
       COUNT(s.vehicle_id) AS total_stolen,
       ROUND(COUNT(s.vehicle_id) * 100.0 / (SELECT COUNT(*) FROM stolen_vehicles), 2) AS percentage
FROM stolen_vehicles s
JOIN make_details m ON s.make_id = m.make_id
GROUP BY m.make_name
ORDER BY total_stolen DESC;
"""

df = pd.read_sql(query, engine)
df
```

Out[52]:

	make_name	total_stolen	percentage
0	Toyota	716	15.78
1	Trailer	543	11.97
2	Nissan	482	10.62
3	Mazda	433	9.54
4	Ford	312	6.88
...	...	...	...
133	Toko	1	0.02
134	Nissan Diesel	1	0.02
135	Alpha	1	0.02
136	Niu	1	0.02
137	Caterpillar	1	0.02

138 rows × 3 columns

- **Conclusion** – Certain makes contribute heavily to overall thefts.
- **Insights & Actions** – Manufacturers can use this data to enhance security features.

16. Use GROUP_CONCAT to list stolen vehicle colors per region.

In [56]:

```
query = """
SELECT l.region, GROUP_CONCAT(DISTINCT s.color ORDER BY s.color ASC) AS stolen_colors
FROM stolen_vehicles s
```

```
JOIN locations l ON s.location_id = l.location_id
GROUP BY l.region;
"""

df = pd.read_sql(query, engine)
df
```

Out[56]:

	region	stolen_colors
0	Auckland	Black,Blue,Brown,Cream,Gold,Green,Grey,Orange,...
1	Bay of Plenty	Black,Blue,Brown,Cream,Gold,Green,Grey,Orange,...
2	Canterbury	Black,Blue,Brown,Gold,Green,Grey,Orange,Pink,P...
3	Gisborne	Black,Blue,Brown,Gold,Green,Grey,Orange,Purple...
4	Hawke's Bay	Black,Blue,Brown,Gold,Green,Grey,Orange,Purple...
5	ManawatÅ«-Whanganui	Black,Blue,Brown,Gold,Green,Grey,Orange,Purple...
6	Nelson	Black,Blue,Gold,Green,Grey,Red,Silver,White
7	Northland	Black,Blue,Brown,Cream,Gold,Green,Grey,Orange,...
8	Otago	Black,Blue,Cream,Gold,Green,Grey,Purple,Red,Si...
9	Southland	Black,Blue,Gold,Green,Grey,Red,Silver,White
10	Taranaki	Black,Blue,Brown,Gold,Green,Grey,Red,Silver,Wh...
11	Waikato	Black,Blue,Brown,Cream,Gold,Green,Grey,Orange,...
12	Wellington	Black,Blue,Brown,Gold,Green,Grey,Orange,Purple...

- **Conclusion** – Shows color trends for stolen vehicles in each region.
- **Insights & Actions** – Helps in identifying common vehicle colors targeted by thieves.

17. List all vehicles stolen in Auckland after 2015, including make, model year, and color.

In [58]:

```
query = """
SELECT sv.vehicle_id, md.make_name, sv.model_year, sv.color
FROM stolen_vehicles sv
JOIN locations l ON sv.location_id = l.location_id
JOIN make_details md ON sv.make_id = md.make_id
WHERE l.region = 'Auckland' AND sv.model_year > 2015;
"""

df = pd.read_sql(query, engine)
df
```

Out[58]:

	vehicle_id	make_name	model_year	color
0	1	Trailer	2021	Silver
1	3	Trailer	2021	Silver
2	5	Trailer	2018	Silver
3	12	Factory Built	2018	Silver
4	17	Trailer	2021	Silver
...	...	...	...	...
295	4017	Kia	2022	Grey
296	4022	Toyota	2016	Silver
297	4031	Mercedes-Benz	2018	Yellow
298	4189	Suzuki	2021	White
299	4532	Niu	2021	White

300 rows × 4 columns

- **Conclusion** – Newer vehicles are being stolen, possibly due to higher resale value. Specific colors or models may be more targeted.
- **Insights & Actions** – This can help insurance companies adjust risk assessments.

18. Find the percentage of stolen vehicles that were trailers for each year.

In [65]:
```
query = """
SELECT model_year,
```

```
       COUNT(CASE WHEN vehicle_type = 'Trailer' THEN 1 END) * 100.0 / COUNT(*) AS trailer_percentage
FROM stolen_vehicles
GROUP BY model_year
ORDER BY model_year DESC;
"""

df = pd.read_sql(query, engine)
df
```

Out[65]:

	model_year	trailer_percentage
0	2022	36.84211
1	2021	40.54054
2	2020	39.47368
3	2019	41.66667
4	2018	36.27451
...	...	...
58	1962	100.00000
59	1960	66.66667
60	1957	0.00000
61	1943	0.00000
62	1940	100.00000

63 rows × 2 columns

- **Conclusion** – The percentage of stolen vehicles that were trailers varies by year, indicating changing theft patterns.

- **Insights & Actions** – If trailer theft is increasing, law enforcement should enhance security at trailer parking areas. Owners should use GPS trackers and heavy-duty locks. Insurance companies may adjust policies based on rising trailer theft trends. Authorities can analyze why specific years have higher trailer theft rates and take preventive measures.

In [ ]:

```

```