Vehicle Registration Investor Dashboard
Streamlit App Code

```python

import streamlit as st
import pandas as pd
import numpy as np
import altair as alt
import matplotlib.pyplot as plt

st.title("Vehicle Registration Investor Dashboard")

uploaded_file = st.file_uploader("Upload your vehicle registration CSV file", type=["csv"])
if uploaded_file:
    try:
        df = pd.read_csv(uploaded_file)

        # Rename columns to expected names if needed
        rename_map = {
            'registration_date': 'reg_date',
            'Period': 'period',
            'Vehicle Class': 'vehicle_category',
            'Manufacturer': 'manufacturer',
            'TOTAL': 'count'
        }
        df.rename(columns=rename_map, inplace=True)

        # Clean column names and values
        df.columns = df.columns.str.strip()
        for col in ['reg_date', 'period', 'vehicle_category', 'manufacturer']:
            if col in df.columns:
                df[col] = df[col].astype(str).str.strip()

        required_cols = ['reg_date', 'vehicle_category', 'manufacturer', 'count']
        missing_cols = [col for col in required_cols if col not in df.columns]
        if missing_cols:
            st.error(f"Missing required columns in uploaded file: {missing_cols}")
            st.stop()

        # Convert types
        df['reg_date'] = pd.to_datetime(df['reg_date'], errors='
            coerce')
        df = df.dropna(subset=['reg_date', 'vehicle_category', 'manufacturer', 'count']).copy()
        df['count'] = pd.to_numeric(df['count'], errors='coerce')
        df = df.dropna(subset=['count'])

        min_date = df['reg_date'].min()
        max_date = df['reg_date'].max()

        date_range = st.date_input("Select Date Range", [min_date, max_date], min_value=min_date, max_value=max_date)
        if len(date_range) == 2:
            start_date, end_date = date_range
            df = df[(df['reg_date'] >= pd.to_datetime(start_date)) & (df['reg_date'] <= pd.to_datetime(end_date))]

        vehicle_categories = sorted(df['vehicle_category'].unique())
        manufacturers = sorted(df['manufacturer'].unique())

        selected_vehicle_categories = st.multiselect("Select Vehicle Categories", vehicle_categories, default=vehicle_categories)
        selected_manufacturers = st.multiselect("Select Manufacturers", manufacturers, default=manufacturers)

        df_filtered = df[
            (df['vehicle_category'].isin(selected_vehicle_categories)) &
            (df['manufacturer'].isin(selected_manufacturers))
        ]

        if not df_filtered.empty:
            latest_year = df_filtered['reg_date'].dt.year.max()
            previous_year = latest_year - 1

            total_latest = df_filtered[df_filtered['reg_date'].dt.year == latest_year]['count'].sum()
            total_previous = df_filtered[df_filtered['reg_date'].dt.year == previous_year]['count'].sum()
            yoy_growth = ((total_latest - total_previous) / total_previous * 100) if total_previous > 0 else None

            # KPIs side by side
            kpi1, kpi2 = st.columns(2)
            kpi1.metric("Total Registrations (Latest Year)", f"{int(total_latest):,}")
            kpi2.metric("YoY Growth (%)", f"{yoy_growth:.2f}%" if yoy_growth is not None else "N/A")

            # Total Registrations Over Time chart
            st.subheader("Total Registrations Over Time (by Quarter)")
            by_quarter = df_filtered.groupby('period')['count'].sum().reset_index()
            line_chart = alt.Chart(by_quarter).mark_line(point=True
```

```python
            ).encode(
                x='period',
                y=alt.Y('count', title="Total Registrations"),
                tooltip=["period", "count"]
            ).properties(width=700, height=400)
            st.altair_chart(line_chart, use_container_width=True)

            # Registrations by Vehicle Category Over Time
            st.subheader("Registrations by Vehicle Category Over Time")
            by_cat = df_filtered.groupby(['period', 'vehicle_category'])['count'].sum().reset_index()
            cat_chart = alt.Chart(by_cat).mark_line(point=True).encode(
                x='period',
                y='count',
                color='vehicle_category',
                tooltip=["period", "vehicle_category", "count"]
            ).properties(width=700, height=400)
            st.altair_chart(cat_chart, use_container_width=True)

            # Registrations by Manufacturer Over Time
            st.subheader("Registrations by Manufacturer Over Time")
            by_manu = df_filtered.groupby(['period', 'manufacturer'])['count'].sum().reset_index()
            manu_chart = alt.Chart(by_manu).mark_line(point=True).encode(
                x='period',
                y='count',
                color='manufacturer',
                tooltip=["period", "manufacturer", "count"]
            ).properties(width=700, height=400)
            st.altair_chart(manu_chart, use_container_width=True)

            # YoY % Change by Vehicle Category
            st.subheader("YoY % Change by Vehicle Category")
            cat_pivot = df_filtered.pivot_table(index='vehicle_category', columns=df_filtered['reg_date'].dt.year, values='count', aggfunc='sum')
            if latest_year in cat_pivot.columns and previous_year in cat_pivot.columns:
                cat_pivot['YoY % Change'] = (cat_pivot[latest_year] - cat_pivot[previous_year]) / cat_pivot[previous_year] * 100
                st.dataframe(cat_pivot[['YoY % Change']].style.format("{:.2f}%"))
            else:
                st.write("Not enough data for YoY % Change by Vehicle Category")

            # Pie Chart for latest year vehicle category share
            st.subheader("Vehicle Registrations by Category (Latest Year)")
            latest_year_data = df_filtered[df_filtered['reg_date'].dt.year == latest_year]
            by_cat_latest = latest_year_data.groupby('vehicle_category')['count'].sum()
            fig, ax = plt.subplots()
            by_cat_latest.plot.pie(autopct='%1.1f%%', ax=ax, legend=False)
            ax.set_ylabel('')
            st.pyplot(fig)

            # Download filtered data
            st.download_button(
                label="Download filtered data as CSV",
                data=df_filtered.to_csv(index=False),
                file_name='filtered_vehicle_registration.csv',
                mime='text/csv'
            )

            # Expandable data preview
            with st.expander("See raw data table"):
                st.dataframe(df_filtered)

        else:
            st.warning("No data found for selected filters and date range.")

    except Exception as e:
        st.error(f"Error processing file: {e}")
else:
    st.info("Please upload a CSV file to begin.")
```

Listing 1: Full Streamlit Dashboard Code for Vehicle Registration Investor Dashboard