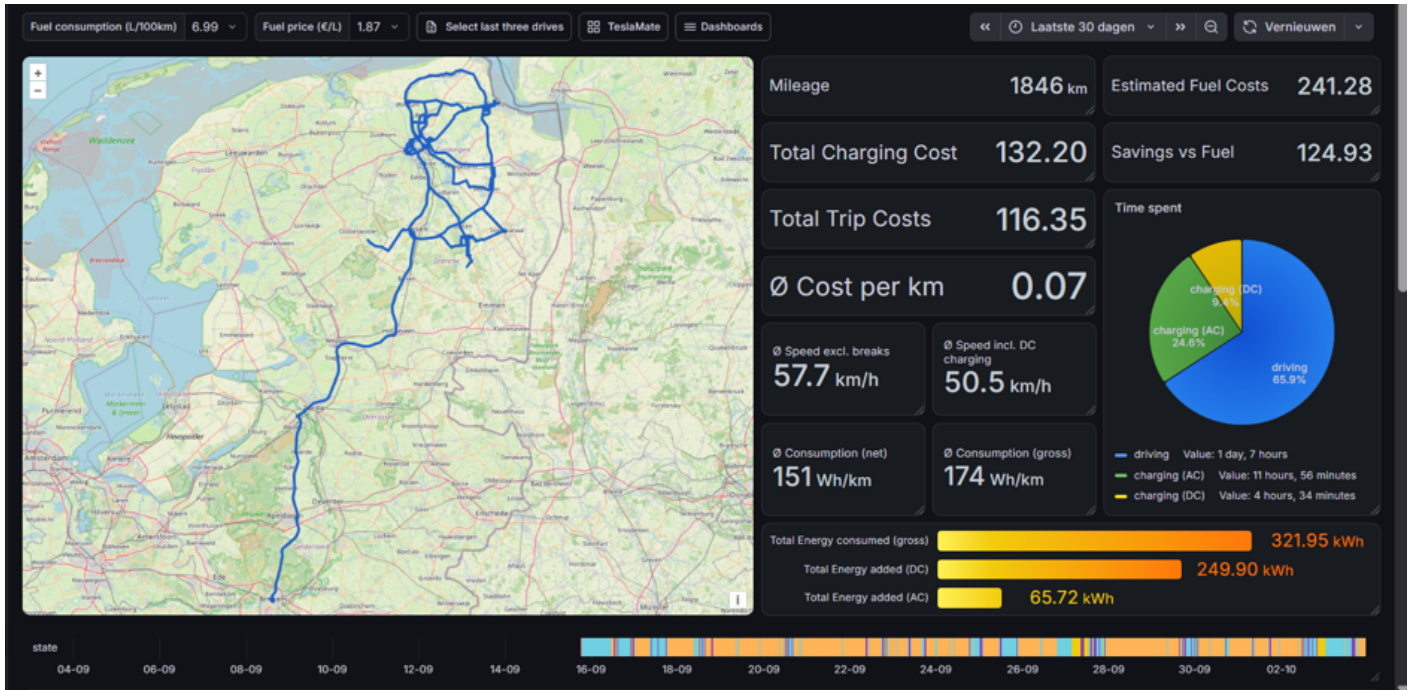


Dashboard configuratie

Documentatie van de aangepaste TeslaMate-dashboards en bijbehorende Grafana-instellingen.



Let op: de SQL-query's in deze handleiding zijn specifiek afgestemd op deze TeslaMate-database en dashboardconfiguratie. Pas query's niet zomaar aan. Bij wijzigingen eerst controleren welke andere dashboards of panelen dezelfde logica gebruiken.

1. Dashboardinstellingen

1.1 Battery Health

Waarden

Instelling	Waarde
Battery Capacity - Usable (new)	49.0 kWh
Ranges - Max range (new)	320.0 km

Deze waarden worden gebruikt als referentie voor de batterijcapaciteit en de maximale actieradius van de auto.

2. Charges

2.1 Estimated Fuel Costs

Doel

Toont wat de gereden afstand naar schatting zou hebben gekost wanneer dezelfde afstand met een benzineauto was gereden.

De berekening gebruikt:

- de gereden afstand;
- het ingestelde benzineverbruik in L/100 km;
- de ingestelde benzineprijs in €/L.

Hiermee kan de laadprijs van de Tesla worden vergeleken met een geschatte brandstofprijs.

Positie

Plaats het paneel:

tussen **Total Charging Cost** en **Duration**.

Query

```
SELECT
  (MAX(odometer) - MIN(odometer)) / 100.0 *
  COALESCE(
```

```

NULLIF(
  REPLACE(
    REGEXP_REPLACE('${fuel_consumption:text}', '^[^0-9.]+', '', 'g'),
    ',', '.',
    ''
  ),
  ''
)::numeric,
6.99
) *
COALESCE(
  NULLIF(
    REGEXP_REPLACE('${fuel_price:text}', '^[^0-9.]+', '', 'g'),
    ''
  ),
  ''
)::numeric,
1.87
) AS "Estimated Fuel Costs"
FROM positions
WHERE
  car_id = $car_id
  AND ideal_battery_range_km IS NOT NULL
  AND (
    drive_id IN (SELECT id FROM drives WHERE $__timeFilter(start_date))
    OR (drive_id IS NULL AND $__timeFilter(date))
  )

```

Variabelen

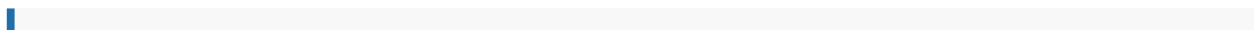
De query gebruikt de dashboardvariabelen:

- `$fuel_consumption`
- `$fuel_price`

Hiermee kan de gebruiker het benzineverbruik en de benzineprijs vanuit het dashboard aanpassen.

De standaardwaarden die in de query als fallback worden gebruikt zijn:

- **6,99 L/100 km**
- **€1,87/L**



Let op: de fallbackwaarden in de query zijn niet automatisch gelijk aan de waarden van de Grafana-variabelen. Wanneer de variabelen worden aangepast, controleer daarom ook of de fallbackwaarden nog gewenst zijn.

3. Charging Stats

3.1 Total Range Added

Doel

Geeft aan hoeveel theoretische actieradius tijdens de geselecteerde laadperiode is toegevoegd.

Positie

Plaats het paneel:

tussen `Total Energy Added` en `Ø Cost per kWh`.

Query

```
SELECT
  convert_km(
    SUM(end_{{preferred_range}}_range_km - start_{{preferred_range}}_range_km)::numeric,
    '{{length_unit}}'
  ) AS "Total Range Added"
FROM
  charging_processes cp
WHERE
  $__timeFilter(end_date)
  AND duration_min >= $min_duration
  AND car_id = $car_id
  AND ('${geofence:pipe}' = '-1' OR cp.geofence_id in ($geofence));
```

3.2 Ø Cost per km (charging stats)

Doel

Berekent de gemiddelde laadkosten per gereden kilometer.

De berekening combineert:

1. het geschatte energieverbruik van de gereden kilometers;
2. de gemiddelde laadprijs per kWh;
3. de totale gereden afstand.

De uitkomst wordt weergegeven als kosten per kilometer.

Positie

Plaats het paneel tussen `Ø Cost per 100 km` en `Total Charging Cost`.

Query

“ **Belangrijk:** deze query gebruikt dezelfde onderliggende berekeningslogica als onderdelen van de Charging Stats-, Efficiency-, Statistics- en Trip-dashboards. Wanneer deze gedeelde logica ooit wordt aangepast, controleer dan ook de andere panelen waarin dezelfde berekening voorkomt.

```
-- Query shared between Charging Stats, Efficiency, Statistics & Trip Dashboards (with minor changes) - ensure to modify in all places when necessary
```

```
with drives_start_event as (
```

```
  select
```

```
    'drive_start' as event, start_date as date, start_${preferred_range}_range_km as  
range, start_km as odometer, car_id, distance is null as is_incomplete
```

```
  from drives
```

```
  where car_id = $car_id and $__timeFilter(start_date) and 48 <= (($__to:date:seconds) -
```

```

$__from:date:seconds}))::numeric / 3600)

),

drives_end_event as (

    select
        'drive_end' as event, case when end_date is null then start_date + interval '1 second'
else end_date end as date, end_{$preferred_range}_range_km as range, end_km as odometer,
car_id, distance is null as is_incomplete
    from drives
    where car_id = $car_id and $__timeFilter(start_date) and 48 <= (($__to:date:seconds} -
$__from:date:seconds}))::numeric / 3600)

),

charging_processes_start_event as (

    select
        'charging_process_start' as event, start_date as date,
start_{$preferred_range}_range_km as range, p.odometer, cp.car_id, end_date is null as
is_incomplete
    from charging_processes cp
        inner join positions p on cp.position_id = p.id
    where cp.car_id = $car_id and $__timeFilter(end_date) and 48 <= (($__to:date:seconds} -
$__from:date:seconds}))::numeric / 3600)

),

charging_processes_end_event as (

    select
        'charging_process_end' as event, case when end_date is null then start_date + interval
'1 second' else end_date end as date, end_{$preferred_range}_range_km as range, p.odometer,
cp.car_id, end_date is null as is_incomplete
    from charging_processes cp
        inner join positions p on cp.position_id = p.id
    where cp.car_id = $car_id and $__timeFilter(end_date) and 48 <= (($__to:date:seconds} -
$__from:date:seconds}))::numeric / 3600)

```

```

),

positions as (

    select
        case
            when drive_id is not null and lead(drive_id) over w is not null then 'drive_start'
            else 'something'
        end as event,
        date, ${preferred_range}_battery_range_km as range, p.odometer, p.car_id, false as
is_incomplete
    from positions p
    where ideal_battery_range_km is not null and car_id = $car_id and 48 >
(($__to:date:seconds} - $__from:date:seconds}):numeric / 3600)
    and (drive_id in (select id from drives where $__timeFilter(start_date)) or drive_id is
null and $__timeFilter(date))
    window w as (order by date)

),

combined as (

    select * from drives_start_event
    union all
    select * from drives_end_event
    union all
    select * from charging_processes_start_event
    union all
    select * from charging_processes_end_event
    union all
    select * from positions

),

final as (

    select
        car_id,
        case when is_incomplete then 0 else lead(odometer) over w - odometer end as distance,
        case when is_incomplete then 0 else case when event != 'drive_start' then

```

```

greatest(range - lead(range) over w, 0) else range - lead(range) over w end end as range_loss
    from combined
    window w as (order by date asc)

),

derived as (

    select
        convert_km(sum(distance)::numeric, '$length_unit') as distance,
        sum(range_loss) * c. efficiency as consumption
    from final
        inner join cars c on car_id = c.id
    group by c. efficiency

),

charges as (

    SELECT
        sum(cost) / sum(charge_energy_added) as cost_per_kwh
    FROM charging_processes cp
    where cp.car_id = $car_id and $__timeFilter(end_date)
        and ('${geofence:pipe}' = '-1' OR cp.geofence_id in ($geofence))

)

select
    consumption / distance * cost_per_kwh as cost_per_km
from derived cross join charges

```

3.3 Top Charging Stations (Cheapest)

Doel

Toont de laadlocaties met de laagste gemiddelde laadkosten per kWh.

“ **Belangrijk:** dit is **niet de goedkoopste individuele laadsessie**. De prijs wordt per locatie berekend als een gewogen gemiddelde op basis van de totale geladen energie. Een locatie met meerdere sessies wordt dus als één gemiddelde weergegeven.

Query

```
SELECT
  COALESCE(
    geofence.name,
    CONCAT_WS(
      ' ',
      COALESCE(address.name, NULLIF(CONCAT_WS(' ', address.road, address.house_number), '')),
      COALESCE(address.city, address.neighbourhood)
    )
  ) AS location,
  SUM(cost) / NULLIF(SUM(GREATEST(charge_energy_added, charge_energy_used)), 0) AS cost_per_kwh
FROM
  charging_processes c
  LEFT JOIN addresses address ON c.address_id = address.id
  LEFT JOIN geofences geofence ON geofence_id = geofence.id
WHERE
  $__timeFilter(end_date)
  AND duration_min >= $min_duration
  AND car_id = $car_id
  AND cost IS NOT NULL
  AND ('${geofence:pipe}' = '-1' OR c.geofence_id IN ($geofence))
GROUP BY
  1
HAVING
  SUM(GREATEST(charge_energy_added, charge_energy_used)) > 0
ORDER BY
  2 ASC NULLS LAST
LIMIT 17;
```

4. Efficiency

4.1 Temperature – Driving Efficiency

Doel

Toont het energieverbruik en de efficiëntie van de auto per buitentemperatuur.

De temperatuur kan worden gegroepeerd in stappen van:

- 1 °C;
- 5 °C;
- 10 °C.

Hierdoor kunnen kleine temperatuurverschillen worden samengevoegd en ontstaat bij grotere datasets een beter leesbare grafiek.

Temperatuurstap

De gebruikte variabele is:

Instelling	Waarde
Type	Custom
Naam	temp_step
Label	Temp step
Display	Above dashboard
Waarden	1, 5, 10

Query

```
WITH t AS (  
  SELECT
```

```

CASE
    WHEN '$temp_unit' = 'C' THEN ROUND(cast(outside_temp_avg AS numeric) / $temp_step, 0) *
$temp_step
    WHEN '$temp_unit' = 'F' THEN ROUND(cast(convert_celsius(outside_temp_avg, '$temp_unit')
AS numeric) / $temp_step, 0) * $temp_step
    END AS outside_temp,
    sum(start_ideal_range_km - end_ideal_range_km) AS total_ideal_range,
    sum(start Rated_range_km - end Rated_range_km) AS total Rated_range,
    sum(distance) AS total_distance,
    sum(duration_min) as duration,
    car_id
FROM
    drives
WHERE
    distance IS NOT NULL
    AND car_id = $car_id
    AND convert_km(distance::numeric, '$length_unit') >= $min_distance
    AND $__timeFilter(start_date)
    AND start_${preferred_range}_range_km - end_${preferred_range}_range_km > 0.1
GROUP BY
    1,
    car_id
)

SELECT
    outside_temp as outside_temp_$temp_unit,
    total_distance / total_${preferred_range}_range AS efficiency,
    total_${preferred_range}_range / convert_km(total_distance::numeric, '$length_unit') *
c.efficiency * 1000 AS consumption_$length_unit,
    convert_km(total_distance::numeric, '$length_unit') as total_distance_$length_unit,
    (convert_km(total_distance::numeric, '$length_unit') / duration) * 60 as
avg_speed_$length_unit
FROM
    t
JOIN cars c ON t.car_id = c.id
WHERE outside_temp IS NOT NULL
ORDER BY 1 DESC

```

5. Statistics

5.1 Statistics

Doel

Dit paneel toont de statistieken per geselecteerde periode.

Standaard worden bij weinig ritten en wanneer er geen laadgegevens beschikbaar zijn niet alle gemiddelden berekend.

De query is daarom aangepast zodat ontbrekende gegevens waar mogelijk worden aangevuld met een berekende waarde op basis van het gereden energieverbruik.

“ **Let op:** deze aanpassing is specifiek bedoeld om ook bij weinig ritten of zonder directe laadgegevens bruikbare statistieken te tonen. Hierdoor kunnen sommige waarden een berekende schatting zijn in plaats van rechtstreeks gemeten laadgegevens.

Query

```
WITH periods AS (  
  SELECT  
    CASE  
      WHEN '$period' = 'total' THEN TIMESTAMPTZ 'epoch'  
      ELSE date_trunc('$period', timezone('UTC', start_date), '$__timezone')  
    END AS date,  
    SUM(GREATEST(start_${preferred_range}_range_km - end_${preferred_range}_range_km, 0)) AS  
range_diff  
  FROM drives  
  WHERE  
    car_id = $car_id  
    AND $__timeFilter(start_date)  
  GROUP BY 1
```

```
),
```

```
charges AS (
```

```
    SELECT
```

```
        CASE
```

```
            WHEN '$period' = 'total' THEN TIMESTAMPTZ 'epoch'
```

```
            ELSE date_trunc('$period', timezone('UTC', start_date), '$__timezone')
```

```
        END AS date,
```

```
        SUM(GREATEST(charge_energy_added, charge_energy_used)) AS sum_energy_used_kwh,
```

```
        SUM(charge_energy_added) AS sum_energy_added_kwh,
```

```
        SUM(GREATEST(charge_energy_added, charge_energy_used)) / COUNT(*) AS
```

```
avg_energy_charged_kwh,
```

```
        SUM(cost) AS cost_charges,
```

```
        COUNT(*) AS cnt_charges
```

```
    FROM charging_processes
```

```
    WHERE
```

```
        car_id = $car_id
```

```
        AND $__timeFilter(start_date)
```

```
        AND (charge_energy_added IS NULL OR charge_energy_added > 0)
```

```
    GROUP BY 1
```

```
),
```

```
avg_cost AS (
```

```
    SELECT
```

```
        SUM(cost) / NULLIF(SUM(GREATEST(charge_energy_added, charge_energy_used)), 0) AS
```

```
cost_per_kwh
```

```
    FROM charging_processes
```

```
    WHERE
```

```
        car_id = $car_id
```

```
        AND $__timeFilter(start_date)
```

```
        AND (charge_energy_added IS NULL OR charge_energy_added > 0)
```

```
        AND cost IS NOT NULL
```

```
),
```

```
car AS (
```

```
    SELECT efficiency
```

```
    FROM cars
```

```
    WHERE id = $car_id
```

```
)
```

```

SELECT
    CASE WHEN '$period' = 'total' THEN $__from ELSE EXTRACT(EPOCH FROM p.date) * 1000 END AS
date_from,
    CASE
        WHEN '$period' = 'total' THEN $__to
        ELSE EXTRACT(EPOCH FROM timezone('$_timezone', timezone('$_timezone', p.date) + ('1 ' ||
'$period')::interval)) * 1000
    END AS date_to,
    CASE '$period'
        WHEN 'total' THEN 'Total'
        WHEN 'month' THEN to_char(timezone('$_timezone', p.date), 'YYYY Month')
        WHEN 'year' THEN to_char(timezone('$_timezone', p.date), 'YYYY')
        WHEN 'week' THEN 'week ' || to_char(timezone('$_timezone', p.date), 'WW') || ' starting '
|| to_char(timezone('$_timezone', p.date), 'YYYY-MM-DD')
        ELSE to_char(timezone('$_timezone', p.date), 'YYYY-MM-DD')
    END AS display,
    p.date,

    COALESCE(c.sum_energy_used_kwh, p.range_diff * car.efficiency) AS sum_energy_used_kwh,
    COALESCE(c.sum_energy_added_kwh, p.range_diff * car.efficiency) AS sum_energy_added_kwh, --
nodig tegen ∞
    c.avg_energy_charged_kwh, -- blijft leeg zonder charge
    COALESCE(c.cost_charges, p.range_diff * car.efficiency * avg_cost.cost_per_kwh) AS
cost_charges,
    c.cnt_charges

FROM periods p
LEFT JOIN charges c ON c.date = p.date
CROSS JOIN avg_cost
CROSS JOIN car

```

6. Trip

6.1 Ø Cost per km (trip)

Doel

Geeft de gemiddelde kosten per gereden kilometer voor de geselecteerde trip/periode.

Dit paneel kan worden gebruikt in plaats van **Ø Cost per 100 km**.

Voor een individuele trip is **kosten per kilometer** vaak logischer dan kosten per 100 km, omdat de daadwerkelijke rit centraal staat.

De query gebruikt de volgende volgorde voor het bepalen van de laadprijs:

1. gemiddelde €/kWh binnen de geselecteerde periode;
2. gemiddelde €/kWh van alle beschikbare laadgegevens;
3. een vaste fallback van **€0,38/kWh**.

Hierdoor blijft het paneel bruikbaar wanneer er in de geselecteerde periode geen laadsessie aanwezig is.

Query

```
WITH charges AS (  
  SELECT  
    COALESCE(  
      -- 1. €/kWh in de geselecteerde periode  
      SUM(cost) FILTER (WHERE $__timeFilter(start_date))  
      / NULLIF(SUM(charge_energy_added) FILTER (WHERE $__timeFilter(start_date)), 0),  
  
      -- 2. overall €/kWh (buiten de periode)  
      SUM(cost) / NULLIF(SUM(charge_energy_added), 0),  
  
      -- 3. vaste fallback  
      0.38  
    ) AS cost_per_kwh  
  FROM charging_processes  
  WHERE  
    car_id = $car_id  
    AND cost IS NOT NULL  
    AND cost > 0  
)
```

```

mileage AS (
  SELECT
    convert_km((MAX(odometer) - MIN(odometer))::numeric, '$length_unit') AS distance
  FROM positions
  WHERE
    car_id = $car_id
    AND ideal_battery_range_km IS NOT NULL
    AND (
      drive_id IN (SELECT id FROM drives WHERE $__timeFilter(start_date))
      OR (drive_id IS NULL AND $__timeFilter(date))
    )
)

SELECT
  'Total Energy consumed (gross)' AS metric,
  cost_per_kwh / NULLIF(distance, 0) AS cost_mileage
FROM mileage
CROSS JOIN charges

```

Transformation

Transformation	Alias
Transformation	Alias
Alias	Ø Cost per \$length_unit

7. Estimated Fuel Costs

7.1 Query

Deze berekening is dezelfde basisberekening als het paneel **Estimated Fuel Costs** binnen Charges, maar wordt hier toegepast op het betreffende dashboard/paneel.

```

SELECT
  (MAX(odometer) - MIN(odometer)) / 100.0 *

```

```

COALESCE(
  NULLIF(
    REPLACE(
      REGEXP_REPLACE('${fuel_consumption:text}', '^[^0-9.]+', '', 'g'),
      ',',
      '.'
    ),
    ''
  )::numeric,
  6.99
) *
COALESCE(
  NULLIF(
    REGEXP_REPLACE('${fuel_price:text}', '^[^0-9.]+', '', 'g'),
    ''
  )::numeric,
  1.87
) AS "Estimated Fuel Costs"
FROM positions
WHERE
  car_id = $car_id
  AND ideal_battery_range_km IS NOT NULL
  AND (
    drive_id IN (SELECT id FROM drives WHERE $__timeFilter(start_date))
    OR (drive_id IS NULL AND $__timeFilter(date))
  )

```

7.2 Eenheid instellen

Wanneer het resultaat als eurobedrag moet worden weergegeven:

1. Verwijder de bestaande override.
2. Voeg een nieuwe override toe.
3. Selecteer het veld:

Estimated Fuel Costs

4. Stel de eenheid in op:

Euro (€).

Override

Instelling	Waarde
Velden met naam	Estimated Fuel Costs
Overschrijvende eigenschap	Eenheid
Eenheid	Euro

8. Dashboardvariabelen

De brandstofberekeningen gebruiken twee variabelen.

8.1 Brandstofprijs

Instelling	Waarde
Type	Custom (of Text box)
Name	fuel_price
Label	Fuel price (€/L)
Values	1.87, 2.00, 2.20, 2.40, 2.411, 2.60
Default	1.87

De waarde bepaalt de gebruikte benzineprijs in euro per liter.

8.2 Brandstofverbruik

Instelling	Waarde
Type	Custom (of Text box)
Name	fuel_consumption
Label	Fuel consumption (L/100km)
Values	6.0, 6.99, 7.0, 8.0
Default	6.99

De waarde bepaalt het gebruikte brandstofverbruik in liter per 100 kilometer.

9. Total Trip Costs

Doel

Berekent de totale geschatte energiekosten van alle ritten binnen de geselecteerde periode.

Wanneer voor een rit voldoende nauwkeurige batterijgegevens beschikbaar zijn, wordt het werkelijke geschatte energieverbruik van die rit gebruikt.

Wanneer de gegevens onvoldoende nauwkeurig zijn, wordt het energieverbruik geschat op basis van het gemiddelde energieverbruik van de ritten waarvoor wel voldoende nauwkeurige gegevens beschikbaar zijn.

Query

```
WITH charges AS (  
  SELECT  
    COALESCE(  
      -- 1. €/kWh in de geselecteerde periode  
      SUM(cost) FILTER (WHERE $__timeFilter(start_date))  
      / NULLIF(SUM(charge_energy_added) FILTER (WHERE $__timeFilter(start_date)), 0),  
  
      -- 2. fallback: overall €/kWh (alle charges met cost)  
      SUM(cost) / NULLIF(SUM(charge_energy_added), 0),  
  
      -- 3. laatste fallback  
      0.37  
    ) AS cost_per_kwh  
  FROM charging_processes  
  WHERE  
    car_id = $car_id  
    AND cost IS NOT NULL  
    AND cost > 0  
) ,  
  
data AS (  
  SELECT  
    drives.distance,
```

```

start_${preferred_range}_range_km - end_${preferred_range}_range_km AS range_diff,
car.efficiency AS car_efficiency,
duration_min > 1
    AND drives.distance > 1
    AND (
        start_position.usable_battery_level IS NULL
        OR end_position.usable_battery_level IS NULL
        OR (end_position.battery_level - end_position.usable_battery_level) = 0
    ) AS is_sufficiently_precise
FROM drives
LEFT JOIN positions start_position
    ON drives.start_position_id = start_position.id
LEFT JOIN positions end_position
    ON drives.end_position_id = end_position.id
LEFT JOIN cars car
    ON drives.car_id = car.id
WHERE
    drives.car_id = $car_id
    AND $__timeFilter(drives.start_date)
),

```

```

stats AS (
    SELECT
        SUM(
            CASE
                WHEN is_sufficiently_precise
                THEN range_diff * car_efficiency
            END
        ) AS total_energy,
        SUM(
            CASE
                WHEN is_sufficiently_precise
                THEN convert_km(distance::numeric, '$length_unit')
            END
        ) AS total_distance
    FROM data
)

```

```

SELECT
    SUM(

```

```

CASE
  WHEN d.is_sufficiently_precise THEN
    d.range_diff * d.car_efficiency * c.cost_per_kwh
  ELSE
    convert_km(d.distance::numeric, '$length_unit') / 100.0
    * (
      s.total_energy
      / NULLIF(s.total_distance, 0)
      * 100.0
      * c.cost_per_kwh
    )
END
) AS "Total Trip Costs"
FROM data d
CROSS JOIN charges c
CROSS JOIN stats s;

```

10. Savings vs Fuel

Doel

Vergelijkt de geschatte kosten van de Tesla met de geschatte brandstofkosten van een benzineauto.

De berekening is:

Estimated Fuel Costs – Total Trip Costs = Savings vs Fuel

Een positief resultaat betekent dat de berekende brandstofkosten hoger zijn dan de berekende Tesla-kosten.

Query

```

WITH charges AS (
  SELECT
    COALESCE(
      SUM(cost) FILTER (WHERE $__timeFilter(start_date))

```

```

        / NULLIF(SUM(charge_energy_added) FILTER (WHERE $__timeFilter(start_date)), 0),
        SUM(cost) / NULLIF(SUM(charge_energy_added), 0),
        0.30
    ) AS cost_per_kwh
FROM charging_processes
WHERE
    car_id = $car_id
    AND cost IS NOT NULL
    AND cost > 0
),

```

```

data AS (
    SELECT
        drives.distance,
        start_${preferred_range}_range_km - end_${preferred_range}_range_km AS range_diff,
        car.efficiency AS car_efficiency,
        duration_min > 1
        AND drives.distance > 1
        AND (
            start_position.usable_battery_level IS NULL
            OR end_position.usable_battery_level IS NULL
            OR (end_position.battery_level - end_position.usable_battery_level) = 0
        ) AS is_sufficiently_precise
    FROM drives
    LEFT JOIN positions start_position
        ON drives.start_position_id = start_position.id
    LEFT JOIN positions end_position
        ON drives.end_position_id = end_position.id
    LEFT JOIN cars car
        ON drives.car_id = car.id
    WHERE
        drives.car_id = $car_id
        AND $__timeFilter(drives.start_date)
),

```

```

stats AS (
    SELECT
        SUM(
            CASE WHEN is_sufficiently_precise
                THEN range_diff * car_efficiency

```

```

END
) AS total_energy,
SUM(
CASE WHEN is_sufficiently_precise
THEN convert_km(distance::numeric, '$length_unit')
END
) AS total_distance
FROM data
),

trip_costs AS (
SELECT
SUM(
CASE
WHEN d.is_sufficiently_precise THEN
d.range_diff * d.car_efficiency * c.cost_per_kwh
ELSE
convert_km(d.distance::numeric, '$length_unit') / 100.0
* (
s.total_energy
/ NULLIF(s.total_distance, 0)
* 100.0
* c.cost_per_kwh
)
)
END
) AS total_trip_costs
FROM data d
CROSS JOIN charges c
CROSS JOIN stats s
),

fuel_costs AS (
SELECT
(MAX(odometer) - MIN(odometer)) / 100.0 *
COALESCE(
NULLIF(
REPLACE(
REGEXP_REPLACE('${fuel_consumption:text}', '^[^0-9.]+', '', 'g'),
',',
'.'
)

```

```

        ),
        ''

)::numeric,
6.99
) *
COALESCE(
    NULLIF(
        REGEXP_REPLACE('${fuel_price:text}', '^[0-9.]', '', 'g'),
        ''
    )::numeric,
    1.83
) AS estimated_fuel_costs
FROM positions
WHERE
    car_id = $car_id
    AND ideal_battery_range_km IS NOT NULL
    AND (
        drive_id IN (SELECT id FROM drives WHERE $__timeFilter(start_date))
        OR (drive_id IS NULL AND $__timeFilter(date))
    )
)

SELECT
    COALESCE(f.estimated_fuel_costs, 0) - COALESCE(t.total_trip_costs, 0) AS "Savings vs Fuel"
FROM fuel_costs f
CROSS JOIN trip_costs t;

```

11. Drives

11.1 Extra kolommen

Aan het **Drives**-paneel zijn drie extra kolommen toegevoegd:

1. **Trip cost**
2. **Estimated Fuel Costs**
3. **Savings vs Fuel**

Hiermee kan per afzonderlijke rit worden bekeken:

- wat de geschatte energiekosten van de Tesla waren;
- wat dezelfde rit met de ingestelde benzineauto zou hebben gekost;
- het verschil tussen beide.

Query

```
WITH cost AS (  
  SELECT  
    COALESCE(  
      SUM(cost) FILTER (WHERE $__timeFilter(start_date))  
      / NULLIF(SUM(charge_energy_added) FILTER (WHERE $__timeFilter(start_date)), 0),  
      SUM(cost) / NULLIF(SUM(charge_energy_added), 0),  
      0.30  
    ) AS cost_per_kwh  
  FROM charging_processes  
  WHERE  
    car_id = $car_id  
    AND cost IS NOT NULL  
    AND cost > 0  
,  
  
data AS (  
  SELECT  
    floor(extract(epoch FROM start_date)) * 1000 AS start_date_ts,  
    ceil(extract(epoch FROM end_date)) * 1000 AS end_date_ts,  
    car.id as car_id,  
    CASE WHEN start_geofence.id IS NULL THEN CONCAT('new?lat=', start_position.latitude,  
'&lng=', start_position.longitude)  
      WHEN start_geofence.id IS NOT NULL THEN CONCAT(start_geofence.id, '/edit')  
    END as start_path,  
    CASE WHEN end_geofence.id IS NULL THEN CONCAT('new?lat=', end_position.latitude, '&lng=',  
end_position.longitude)  
      WHEN end_geofence.id IS NOT NULL THEN CONCAT(end_geofence.id, '/edit')  
    END as end_path,  
    TO_CHAR((duration_min * INTERVAL '1 minute'), 'HH24:MI') as duration_str,  
    drives.id as drive_id,  
    start_date,  
    COALESCE(start_geofence.name, CONCAT_WS(' ', COALESCE(start_address.name,
```

```

nullif(CONCAT_WS(' ', start_address.road, start_address.house_number), '')),
COALESCE(start_address.city, start_address.neighbourhood))) AS start_address,
    COALESCE(end_geofence.name, CONCAT_WS(' ', COALESCE(end_address.name, nullif(CONCAT_WS('
', end_address.road, end_address.house_number), '')), COALESCE(end_address.city,
end_address.neighbourhood))) AS end_address,
    duration_min,
    distance,
    start_position.usable_battery_level as start_usable_battery_level,
    start_position.battery_level as start_battery_level,
    end_position.usable_battery_level as end_usable_battery_level,
    end_position.battery_level as end_battery_level,
    start_position.battery_level != start_position.usable_battery_level OR
end_position.battery_level != end_position.usable_battery_level as reduced_range,
    duration_min > 1 AND distance > 1 AND (
        start_position.usable_battery_level IS NULL OR end_position.usable_battery_level IS NULL
OR
        (end_position.battery_level - end_position.usable_battery_level) = 0
    ) as is_sufficiently_precise,
    start_{$preferred_range}_range_km - end_{$preferred_range}_range_km as range_diff,
    car.efficiency as car_efficiency,
    outside_temp_avg,
    distance / NULLIF(duration_min, 0) * 60 AS avg_speed
FROM drives
LEFT JOIN addresses start_address ON start_address_id = start_address.id
LEFT JOIN addresses end_address ON end_address_id = end_address.id
LEFT JOIN positions start_position ON start_position_id = start_position.id
LEFT JOIN positions end_position ON end_position_id = end_position.id
LEFT JOIN geofences start_geofence ON start_geofence_id = start_geofence.id
LEFT JOIN geofences end_geofence ON end_geofence_id = end_geofence.id
LEFT JOIN cars car ON car.id = drives.car_id
WHERE $__timeFilter(start_date) AND drives.car_id = $car_id
ORDER BY start_date DESC
)

SELECT
    start_date_ts,
    end_date_ts,
    car_id,
    start_path,
    end_path,

```

```

duration_str,
drive_id,
start_date,
start_address,
end_address,
duration_min,
convert_km(distance::numeric, '$length_unit') AS distance_$length_unit,
start_battery_level as "% Start",
end_battery_level as "% End",
CASE WHEN is_sufficiently_precise THEN range_diff * car_efficiency /
convert_km(distance::numeric, '$length_unit') * 1000
END AS consumption_kWh_$length_unit,

-- Trip cost (energy)
CASE WHEN is_sufficiently_precise THEN
    GREATEST(range_diff, 0) * car_efficiency * cost.cost_per_kwh
END AS "Trip cost",

-- Estimated Fuel Costs
distance / 100.0 *
COALESCE(
    NULLIF(
        REPLACE(
            REGEXP_REPLACE('${fuel_consumption:text}', '^[^0-9.]+', '', 'g'),
            ',',
            '.'
        ),
        ''
    )::numeric,
    6.99
) *
COALESCE(
    NULLIF(
        REGEXP_REPLACE('${fuel_price:text}', '^[^0-9.]+', '', 'g'),
        ''
    )::numeric,
    1.87
) AS "Estimated Fuel Costs",

-- Savings vs Fuel

```

```

(
  distance / 100.0 *
  COALESCE(
    NULLIF(
      REPLACE(
        REGEXP_REPLACE('${fuel_consumption:text}', '^[^0-9.]+', '', 'g'),
        ',',
        '.'
      ),
      ''
    )::numeric,
    6.99
  ) *
  COALESCE(
    NULLIF(
      REGEXP_REPLACE('${fuel_price:text}', '^[^0-9.]+', '', 'g'),
      ''
    )::numeric,
    1.87
  )
)
-
COALESCE(
  CASE WHEN is_sufficiently_precise THEN
    GREATEST(range_diff, 0) * car_efficiency * cost.cost_per_kwh
  END,
  0
) AS "Savings vs Fuel"

FROM data
CROSS JOIN cost;

```

12. Dashboardvolgorde

De elementen worden in onderstaande volgorde weergegeven.

12.1 Kosten en vergelijking

1. **Mileage**
2. **Estimated Fuel Costs**
3. **Total Charging Cost**
4. **Savings vs Fuel**
5. **Total Trip Costs**

12.2 Rijtijd en efficiëntie

6. **Time spent (driving + charging)**
 7. **Ø Cost per km**
 8. **Ø Speed excl. breaks**
 9. **Ø Speed incl. DC charging**
 10. **Ø Consumption (net)**
 11. **Ø Consumption (gross)**
-

13. Belangrijke aandachtspunten

13.1 Queries niet los van elkaar aanpassen

Een aantal berekeningen gebruikt dezelfde uitgangspunten, maar is niet noodzakelijk letterlijk dezelfde query.

Vooraf bij:

- Ø Cost per km
- Total Trip Costs
- Savings vs Fuel
- Drives

moet bij een wijziging worden gecontroleerd of dezelfde berekeningslogica ook elders moet worden aangepast.

13.2 Laadkosten

Wanneer binnen de geselecteerde periode geen bruikbare laadprijs beschikbaar is, gebruiken verschillende berekeningen een fallback:

€0,38/kWh

Dit voorkomt dat panelen leeg blijven wanneer er geen laadsessie met kosten beschikbaar is.

13.3 Brandstofvergelijking

De brandstofvergelijking is een **schatting**.

De uitkomst is afhankelijk van:

- ingesteld brandstofverbruik;
- ingestelde benzineprijs;
- gereden afstand;
- beschikbare TeslaMate-data;
- gebruikte laadprijs.

De waarde moet daarom worden gezien als een vergelijkingswaarde en niet als een exact bedrag dat daadwerkelijk met een benzineauto zou zijn uitgegeven.

13.4 Eenheden

De dashboards maken gebruik van TeslaMate/Grafana-variabelen zoals:

- `$length_unit`
- `$temp_unit`
- `$preferred_range`
- `$car_id`
- `$geofence`
- `$min_duration`
- `$min_distance`

Wanneer deze variabelen of de onderliggende TeslaMate-configuratie veranderen, kunnen meerdere panelen hierdoor worden beïnvloed.

14. Wijzigingslog

Gebruik deze sectie om toekomstige wijzigingen aan het dashboard bij te houden.

Datum	Wijziging	Opmerking
2026-10-03	Documentatie opgeschoond	Structuur, spelling en toelichtingen bijgewerkt

15. Snelle referentie

Paneel	Functie
Estimated Fuel Costs	Geschatte benzinekosten voor dezelfde afstand
Total Range Added	Toegevoegde actieradius tijdens laden
Ø Cost per km	Gemiddelde laad-/energiekosten per km
Top Charging Stations	Laadlocaties met laagste gewogen gemiddelde €/kWh
Temperature - Driving Efficiency	Efficiëntie uitgesplitst naar buitentemperatuur
Statistics	Statistieken per periode, inclusief berekende fallbackwaarden
Ø Cost per km (Trip)	Kosten per kilometer voor een trip/periode
Total Trip Costs	Totale geschatte energiekosten van de ritten
Savings vs Fuel	Geschat verschil tussen benzinekosten en Tesla-kosten
Drives - Trip cost	Geschatte energiekosten per individuele rit
Drives - Estimated Fuel Costs	Geschatte benzinekosten per individuele rit
Drives - Savings vs Fuel	Geschat verschil per individuele rit