3. GPS Fleet - Driver Risk and Route Anomaly Detection

Status	In progress

Project Name : GPS Fleet Data - Driver Risk & Route Anomaly Detection →
Alternate Names : DriverRisk-DBRA24, Fleet Anomalies-Snowflake

Business Problem : Build a system that identifies risky drivers and flags anomalous trips so fleet managers can prioritize interventions and reduce incident risk.

What business issues this solves:

- proactive safety monitoring (prevent accidents)
- operational cost savings (less fuel/wear from risky driving)
- compliance & route theft/geofence alerting
- performance-based driver coaching and incentives

Secondary analyses:

1. Route deviation analysis and hotspot mapping (why did deviation happen: traffic, weather, etc.)
2. Driver behavioral consistency score & trend (who's improving/worsening)
3. Clustering of trip types or driver segments (group similar driving patterns)

Created GPSFLEET_WH , FLEET_DB and scehmas name are - raw, clean and analytics.

Currently using FLEET_DB.RAW

snowflake sql file name :: 00.setup_create_db_schema.sql

```
-- warehouse set to the newly created GPSFLEET_WH warehouse
-- creating database
```

```
create or replace database fleet_db

-- creating schemas inside the database
create or replace schema fleet_db.raw;
create or replace schema fleet_db.clean;
create or replace schema fleet_db.analytics

-- type of csv format step and skipping header
use database fleet_db

CREATE OR REPLACE FILE FORMAT csv_fmt TYPE = 'CSV'
  FIELD_OPTIONALLY_ENCLOSED_BY = '"'
  SKIP_HEADER = 1;

-- create stage so that I can load my csv file in this stage.
CREATE OR REPLACE STAGE my_stage
  FILE_FORMAT = csv_fmt;

-- loaded the data

-- check if file loaded in stage or not
list @my_stage

-- create trips_raw table
CREATE OR REPLACE TABLE fleet_db.raw.trips_raw (
  trip_id STRING,
  driver_id STRING,
  vehicle_id STRING,
  timestamp STRING,
  latitude DOUBLE,
  longitude DOUBLE,
  speed DOUBLE,
  acceleration DOUBLE,
  steering_angle DOUBLE,
  heading DOUBLE,
  trip_duration DOUBLE,
  trip_distance DOUBLE,
  fuel_consumption DOUBLE,
  rpm DOUBLE,
  brake_usage DOUBLE,
  lane_deviation DOUBLE,
  weather_conditions STRING,
  road_type STRING,
  traffic_condition STRING,
  stop_events INTEGER,
  geofencing_violation BOOLEAN,
  anomalous_event BOOLEAN,
  route_anomaly BOOLEAN,
  route_deviation_score DOUBLE,
  acceleration_variation DOUBLE,
  behavioral_consistency_index DOUBLE
)


-- copy the data from stage to the trips_raw table created

copy into fleet_db.raw.trips_raw from @my_stage/driver_behavior_route_anomaly_dataset_with_derived_features.csv file_format = (FORMAT_NAME='csv_fmt') ON_ERROR='CONTINUE'

-- check if data copied or not
select * from fleet_db.raw.trips_raw


DROP TABLE FLEET_DB.RAW_TRIPS_RAW

-- changed all data types to string
CREATE OR REPLACE TABLE fleet_db.raw.trips_raw (
  trip_id STRING,
  driver_id STRING,
  vehicle_id STRING,
  timestamp STRING,
  latitude STRING,
  longitude STRING,
  speed STRING,
  acceleration STRING,
  steering_angle STRING,
  heading STRING,
  trip_duration STRING,
  trip_distance STRING,
  fuel_consumption STRING,
  rpm STRING,
  brake_usage STRING,
  lane_deviation STRING,
  weather_conditions STRING,
  road_type STRING,
  traffic_condition STRING,
  stop_events STRING,
  geofencing_violation STRING,
  anomalous_event STRING,
  route_anomaly STRING,
  route_deviation_score STRING,
  acceleration_variation STRING,
  behavioral_consistency_index STRING
)


-- copy the data from stage files to the trips_raw table created
COPY INTO fleet_db.raw.trips_raw FROM @my_stage/driver_behavior_route_anomaly_dataset_with_derived_features.csv FILE_FORMAT=(FORMAT_NAME='csv_fmt') ON_ERROR='CONTINUE';
```

created a new sql file in workspace 01.raw_to_clean_table.sql

```
use database fleet_db

select * from fleet_db.raw.trips_raw
```

```
-- to check the data type of the columns
desc table fleet_db.raw.trips_raw

-- another way to check the data types of the columns
select column_name, data_type from FLEET_DB.INFORMATION_SCHEMA.COLUMNS
where TABLE_NAME = 'TRIPS_RAW' and  TABLE_SCHEMA = 'RAW'

-- check if timestamp format is all good or not
select
    trip_id,
    timestamp
from fleet_db.raw.trips_raw
where try_to_timestamp(timestamp) is null
-- check if the format of latitude and longitude is all good
SELECT trip_id, latitude, longitude
FROM fleet_db.raw.trips_raw
WHERE TRY_TO_DOUBLE(latitude) IS NULL OR TRY_TO_DOUBLE(longitude) IS NULL

-- create trips_clean table by changing all columns to their respectie data types from original trips_raw table
CREATE OR REPLACE TABLE fleet_db.clean.trips_clean AS
SELECT
  TRIM(trip_id) AS trip_id,
  TRIM(driver_id) AS driver_id,
  TRIM(vehicle_id) AS vehicle_id,

  -- safe timestamp parse (NULL if cannot parse)
  TRY_CAST(timestamp AS TIMESTAMP) AS time_stamp,

  TRY_CAST(latitude AS DOUBLE) AS latitude,
  TRY_CAST(longitude AS DOUBLE) AS longitude,
  TRY_CAST(speed AS DOUBLE) AS speed,
  TRY_CAST(acceleration AS DOUBLE) AS acceleration,
  TRY_CAST(steering_angle AS DOUBLE) AS steering_angle,
  TRY_CAST(heading AS DOUBLE) AS heading,
  TRY_CAST(trip_duration AS DOUBLE) AS trip_duration,
  TRY_CAST(trip_distance AS DOUBLE) AS trip_distance,
  TRY_CAST(fuel_consumption AS DOUBLE) AS fuel_consumption,
  TRY_CAST(rpm AS DOUBLE) AS rpm,

  -- brake_usage is a count → integer
  TRY_CAST(brake_usage AS INTEGER) AS brake_usage,

  TRY_CAST(lane_deviation AS DOUBLE) AS lane_deviation,

  LOWER(TRIM(weather_conditions)) AS weather_conditions,
  LOWER(TRIM(road_type)) AS road_type,
  LOWER(TRIM(traffic_condition)) AS traffic_condition,

  TRY_CAST(stop_events AS INTEGER) AS stop_events,

  -- boolean mapping from 1/0 or 'true' strings
  IFF(TRY_CAST(geofencing_violation AS INTEGER) = 1, TRUE, FALSE) AS geofencing_violation,
  IFF(TRY_CAST(anomalous_event AS INTEGER) = 1, TRUE, FALSE) AS anomalous_event,
  IFF(TRY_CAST(route_anomaly AS INTEGER) = 1, TRUE, FALSE) AS route_anomaly,

  TRY_CAST(route_deviation_score AS DOUBLE) AS route_deviation_score,
  TRY_CAST(acceleration_variation AS DOUBLE) AS acceleration_variation,
  TRY_CAST(behavioral_consistency_index AS DOUBLE) AS behavioral_consistency_index

FROM fleet_db.raw.trips_raw;

-- some checks
select * from fleet_db.clean.trips_clean
select count(*) from fleet_db.clean.trips_clean

select
column_name,
data_type
from FLEET_DB.INFORMATION_SCHEMA.COLUMNS
where TABLE_NAME = 'TRIPS_CLEAN' and TABLE_SCHEMA ='CLEAN'

-- some checks
select count(case when time_stamp is null then 1 end) as null_time_stamps from fleet_db.clean.trips_clean
--
select  min(time_stamp), max(time_stamp) from fleet_db.clean.trips_clean
--
select distinct
    weather_conditions,
    road_type,
    traffic_condition
from fleet_db.clean.trips_clean
```

created a new sql file in snowflake workspace 02.analysis_table.sql

```
select * from fleet_db.clean.trips_clean



-- creating trip_summary table for analytics
create or replace table fleet_db.analytics.trip_summary as
select
    trip_id,
    driver_id,
    vehicle_id,
    date_trunc('day', time_stamp) as day,
    max(time_stamp) as trip_start_at,
    max(trip_duration)/60 as trip_duration_minutes,
    max(trip_distance) as trip_distance_Km,
    avg(speed) as avg_speed,
    max(speed) as max_speed,
    avg(acceleration) as avg_accelaration,
    sum(brake_usage) as total_brake_events,
    max(route_deviation_score) as max_route_deviation,
    max(anomalous_event) as any_anomalous_event,
    max(geofencing_violation) as any_geofencing_violation,
    max(route_anomaly) as any_route_anomaly,
    avg(route_deviation_score) as avg_route_deviation_score,
    avg(behavioral_consistency_index) as behavioral_consistency_index
from fleet_db.clean.trips_clean
group by trip_id, driver_id, vehicle_id, date_trunc('day', time_stamp)

-- how many trips had anomalous even and route anomaly events
select * from fleet_db.analytics.trip_summary
where any_anomalous_event = True and any_route_anomaly = True

-- anomalos event and route anomaly events by driver
select
driver_id,
count(distinct trip_id) as total_trips,
SUM(case when any_anomalous_event = True then 1 else 0 end) as number_of_anomalous_event,
SUM(case when any_route_anomaly = True then 1 else 0 end) as number_of_route_anomaly_event,
from fleet_db.analytics.trip_summary
group by driver_id

-- create driver daily table
create or replace table fleet_db.analytics.driver_daily as
select
    driver_id,
    to_date(day) as day,
    count(distinct trip_id) as trip_count_daily,
    avg(trip_distance_Km) as avg_trip_distance_daily,
    avg(avg_speed) as avg_speed_daily,
    sum(total_brake_events) as total_brake_events_daily,
    sum(case when any_anomalous_event = True then 1 else 0 end) as anomalies_count_daily,
    sum(case when any_route_anomaly = True  then 1 else 0 end) as route_anomalies_daily,
    sum(case when any_geofencing_violation = True  then 1 else 0 end) as geofencing_violations_daily,
    avg(max_route_deviation) as avg_route_deviation_daily,
    avg(behavioral_consistency_index) as behavioral_consistency_index_daily
from fleet_db.analytics.trip_summary
group by driver_id, day

select * from fleet_db.analytics.driver_daily order by driver_id asc

-- create vehicle_daily table
create or replace table fleet_db.analytics.vehicle_daily as
select
    vehicle_id,
    to_date(time_stamp) as day,
    count(*) as trips_count,
    avg(trip_distance) as avg_trip_distance_daily,
    avg(fuel_consumption) as avg_fuel_consumption_daily,
    avg(rpm) as avg_rpm_daily,
    avg(speed) as avg_speed_daily
from fleet_db.clean.trips_clean
group by vehicle_id, day

select * from fleet_db.analytics.vehicle_daily order by vehicle_id asc, day asc

-- create route anomalies table
CREATE OR REPLACE TABLE fleet_db.analytics.route_anomalies AS
SELECT
  trip_id,
  driver_id,
  vehicle_id,
  to_date(time_stamp) as day,
  trip_distance,
  speed,
  (trip_duration)/60 as trip_duration_minutes,
  (trip_duration)/3600 as trip_duration_hours,
  route_anomaly,
  anomalous_event,
  geofencing_violation,
  weather_conditions,
  traffic_condition,
  road_type,
  lane_deviation,
  route_deviation_score, -- lower this values the better
  behavioral_consistency_index -- higher this value the better
FROM fleet_db.clean.trips_clean
WHERE route_anomaly = TRUE OR anomalous_event = TRUE

-- Number of trips where the route anomaly, anomalou event s and geofencin
g violations were True by traffic conditions
select
    traffic_condition,
    count(distinct trip_id) as number_of_trips
from fleet_db.analytics.route_anomalies
where route_anomaly = True and anomalous_event = True and geofencing_vio
lation = True
group by traffic_condition

-- create daily_rollup table
CREATE OR REPLACE TABLE fleet_db.analytics.daily_rollup AS
SELECT
  to_date(time_stamp) AS day,
  COUNT(*) AS total_records,
  COUNT(DISTINCT trip_id) AS unique_trips,
  COUNT(DISTINCT driver_id) AS active_drivers,
  COUNT(DISTINCT vehicle_id) AS active_vehicles,
  AVG(speed) AS avg_speed,
  AVG(route_deviation_score) AS avg_route_deviation_score,
  sum(case when anomalous_event = True then 1 else 0 end) as anomalies_co
```

```
unt_daily,
  sum(case when route_anomaly = True  then 1 else 0 end) as route_anomalies_daily,
  sum(case when geofencing_violation = True then 1 else 0 end) as geofencing_violations_daily
FROM fleet_db.clean.trips_clean
GROUP BY day

select * from fleet_db.analytics.daily_rollup

-- create context_summary table
create or replace table fleet_db.analytics.context_summary as
select
    to_date(time_stamp) as day,
    road_type,
    traffic_condition,
    weather_conditions,
    avg(speed) as avg_speed,
    avg(steering_angle) as avg_steering_angle,
    avg(trip_duration) as avg_trip_duration_seconds,
    avg(trip_distance) as avg_trip_distance,
    avg(fuel_consumption) as avg_fuel_consumption,
    avg(lane_deviation) as avg_lane_deviation,
    sum(stop_events) as total_stop_events,
    sum(case when anomalous_event = True then 1 else 0 end) as anomalies_count,
    sum(case when route_anomaly = True  then 1 else 0 end) as route_anomalies,
    sum(case when geofencing_violation = True then 1 else 0 end) as geofencing_violations,
    avg(route_deviation_score) as avg_route_deviation_score,
    avg(behavioral_consistency_index) as avg_driver_behaviour_score
from fleet_db.clean.trips_clean
group by day, road_type, traffic_condition, weather_conditions

select * from fleet_db.analytics.context_summary order by day asc, road_type
```

```
asc;
```

we created a new sql file in workspace 03.analysis_insights.sql

```
select * from fleet_db.analytics.driver_daily

-- drivers and their risk score
WITH stats AS (
  SELECT
    MIN(anomalies_count_daily) AS min_anom,
    MAX(anomalies_count_daily) AS max_anom,
    MIN(route_anomalies_daily) AS min_ran,
    MAX(route_anomalies_daily) AS max_ran,
    MIN(geofencing_violations_daily) AS min_geo,
    MAX(geofencing_violations_daily) AS max_geo,
    MIN(avg_route_deviation_daily) AS min_rdev,
    MAX(avg_route_deviation_daily) AS max_rdev,
    MIN(behavioral_consistency_index_daily) AS min_beh,
    MAX(behavioral_consistency_index_daily) AS max_beh
  FROM fleet_db.analytics.driver_daily
),
norm AS (
  SELECT
    d.driver_id,
    d.day,
    CASE WHEN s.max_anom = s.min_anom THEN 0
         ELSE (d.anomalies_count_daily - s.min_anom) / NULLIF(s.max_anom - s.min_anom,0) END AS anom_n,
    CASE WHEN s.max_ran = s.min_ran THEN 0
         ELSE (d.route_anomalies_daily - s.min_ran) / NULLIF(s.max_ran - s.min_ran,0) END AS ran_n,
    CASE WHEN s.max_geo = s.min_geo THEN 0
         ELSE (d.geofencing_violations_daily - s.min_geo) / NULLIF(s.max_geo - s.min_geo,0) END AS geo_n,
    CASE WHEN s.max_rdev = s.min_rdev THEN 0
```

```
        ELSE (d.avg_route_deviation_daily - s.min_rdev) / NULLIF(s.max_rdev - s.min_rdev,0) END AS rdev_n,
    CASE WHEN s.max_beh = s.min_beh THEN 0
        ELSE 1 - ((d.behavioral_consistency_index_daily - s.min_beh) / NULLIF(s.max_beh - s.min_beh,0)) END AS beh_inv_n
  FROM fleet_db.analytics.driver_daily d CROSS JOIN stats s
)
SELECT
  driver_id,
  day,
  ROUND(
    0.35*beh_inv_n + 0.30*geo_n + 0.20*anom_n + 0.10*ran_n + 0.05*rdev_n
  ,4) AS risk_score_0_1,
 ROUND(
    10 * (0.35*beh_inv_n + 0.30*geo_n + 0.20*anom_n + 0.10*ran_n + 0.05*rdev_n)
  ,3) AS risk_score_1_10,
FROM norm
ORDER BY risk_score_0_1 DESC

-- top 5 risky drivers with their risk scores
WITH stats AS (
  SELECT
    MIN(anomalies_count_daily) AS min_anom,
    MAX(anomalies_count_daily) AS max_anom,
    MIN(route_anomalies_daily) AS min_ran,
    MAX(route_anomalies_daily) AS max_ran,
    MIN(geofencing_violations_daily) AS min_geo,
    MAX(geofencing_violations_daily) AS max_geo,
    MIN(avg_route_deviation_daily) AS min_rdev,
    MAX(avg_route_deviation_daily) AS max_rdev,
    MIN(behavioral_consistency_index_daily) AS min_beh,
    MAX(behavioral_consistency_index_daily) AS max_beh
  FROM fleet_db.analytics.driver_daily
),
norm AS (
  SELECT
    d.driver_id,
    d.day,
    CASE WHEN s.max_anom = s.min_anom THEN 0
         ELSE (d.anomalies_count_daily - s.min_anom) / NULLIF(s.max_anom - s.min_anom,0) END AS anom_n,
    CASE WHEN s.max_ran = s.min_ran THEN 0
         ELSE (d.route_anomalies_daily - s.min_ran) / NULLIF(s.max_ran - s.min_ran,0) END AS ran_n,
    CASE WHEN s.max_geo = s.min_geo THEN 0
         ELSE (d.geofencing_violations_daily - s.min_geo) / NULLIF(s.max_geo - s.min_geo,0) END AS geo_n,
    CASE WHEN s.max_rdev = s.min_rdev THEN 0
         ELSE (d.avg_route_deviation_daily - s.min_rdev) / NULLIF(s.max_rdev - s.min_rdev,0) END AS rdev_n,
    CASE WHEN s.max_beh = s.min_beh THEN 0
         ELSE 1 - ((d.behavioral_consistency_index_daily - s.min_beh) / NULLIF(s.max_beh - s.min_beh,0)) END AS beh_inv_n
  FROM fleet_db.analytics.driver_daily d CROSS JOIN stats s
),
scored as(
select
  driver_id,
  day,
  ROUND(
    0.35*beh_inv_n + 0.30*geo_n + 0.20*anom_n + 0.10*ran_n + 0.05*rdev_n
  ,4) AS risk_score_0_1,
 ROUND(
    10 * (0.35*beh_inv_n + 0.30*geo_n + 0.20*anom_n + 0.10*ran_n + 0.05*rdev_n)
  ,3) AS risk_score_1_10,
  RANK() OVER (ORDER BY 0.35*beh_inv_n + 0.30*geo_n + 0.20*anom_n + 0.10*ran_n + 0.05*rdev_n DESC) AS risk_rank
  FROM norm
)
SELECT *
```

```
FROM scored
WHERE risk_rank <= 5
ORDER BY risk_rank



-- Top risky trips (simple, readable, parameterizable)
-- Simple Trip Risk (flags 0/0.33/0.67/1, severity = avg(norm_route_dev, inv_norm_behavior))
WITH base AS (
  SELECT
    trip_id,
    driver_id,
    day,
    CASE WHEN any_anomalous_event THEN 1 ELSE 0 END AS f_anom,
    CASE WHEN any_route_anomaly THEN 1 ELSE 0 END   AS f_route,
    CASE WHEN any_geofencing_violation THEN 1 ELSE 0 END AS f_geo,
    COALESCE(max_route_deviation,0)                 AS route_dev,
    COALESCE(behavioral_consistency_index,0)        AS beh_idx
  FROM fleet_db.analytics.trip_summary
),
-- global mins / maxs for normalization
stats AS (
  SELECT
    MIN(route_dev) AS min_rdev, MAX(route_dev) AS max_rdev,
    MIN(beh_idx)  AS min_beh,  MAX(beh_idx)  AS max_beh
  FROM base
),
norm AS (
  SELECT
    b.*,
    -- flag count and mapped flag score (your chosen mapping: 0,0.33,0.67,1)
    (f_anom + f_route + f_geo) AS flag_count,
    CASE (f_anom + f_route + f_geo)
      WHEN 0 THEN 0.00
      WHEN 1 THEN 0.33
      WHEN 2 THEN 0.67
      ELSE 1.00
    END AS flag_score,
    -- normalized route deviation 0..1
    CASE WHEN s.max_rdev = s.min_rdev THEN 0
         ELSE (b.route_dev - s.min_rdev) / NULLIF(s.max_rdev - s.min_rdev,0) END AS n_route_dev,
    -- normalized inverted behavior 0..1 (higher = worse)
    CASE WHEN s.max_beh = s.min_beh THEN 0
         ELSE 1 - ((b.beh_idx - s.min_beh) / NULLIF(s.max_beh - s.min_beh,0)) END AS n_beh_inv
  FROM base b CROSS JOIN stats s
),
severity AS (
  SELECT
    *,
    -- severity is simple average of the two normalized signals
    (n_route_dev + n_beh_inv) / 2.0 AS severity_raw
  FROM norm
),
risk_scored as (
  SELECT
    s.trip_id,
    s.driver_id,
    s.day,
    s.flag_count,
    ROUND(s.flag_score,2)         AS flag_score,
    ROUND(s.severity_raw,4)       AS severity_0_1,
    -- raw weighted score (already 0..1)
    ROUND(0.7*s.flag_score + 0.3*s.severity_raw,4) AS risk_score_0_1,
    -- 1..10
    ROUND(10 * (0.7*s.flag_score + 0.3*s.severity_raw),3) AS risk_score_1_10,
    RANK() OVER (ORDER BY 0.7*s.flag_score + 0.3*s.severity_raw DESC) AS risk_rank
  FROM severity s
)
```

```
select *
from risk_scored
where risk_rank <=5
ORDER BY risk_score_0_1 DESC



-- total anomalies per day
SELECT
    day,
    anomalies_count_daily,
    route_anomalies_daily,
    geofencing_violations_daily,
    anomalies_count_daily
      + route_anomalies_daily
      + geofencing_violations_daily AS total_risk_events
FROM fleet_db.analytics.daily_rollup
ORDER BY day;
```

created another sql file in workspace, 04.tables_Bi_ML.sql

```
-- tables to be loaded in Power BI for Dashboard
CREATE OR REPLACE TABLE fleet_db.analytics.trip_map AS
SELECT
  t.trip_id,
  t.driver_id,
  t.day,
  MIN(c.latitude)    AS start_lat,    -- or use first/last depending on desired point
  MIN(c.longitude)   AS start_lon,
  t.max_route_deviation,
  t.any_anomalous_event,
  t.any_route_anomaly,
  t.any_geofencing_violation
FROM fleet_db.analytics.trip_summary t
JOIN fleet_db.clean.trips_clean c USING (trip_id)
GROUP BY t.trip_id, t.driver_id, t.day, t.max_route_deviation,
         t.any_anomalous_event, t.any_route_anomaly, t.any_geofencing_violation;


select * from fleet_db.analytics.trip_map

select * from fleet_db.analytics.daily_rollup

select * from fleet_db.analytics.context_summary


CREATE OR REPLACE TABLE fleet_db.analytics.driver_risk_score AS
WITH stats AS (
  SELECT
    MIN(anomalies_count_daily) AS min_anom,
    MAX(anomalies_count_daily) AS max_anom,
    MIN(route_anomalies_daily) AS min_ran,
    MAX(route_anomalies_daily) AS max_ran,
    MIN(geofencing_violations_daily) AS min_geo,
    MAX(geofencing_violations_daily) AS max_geo,
    MIN(avg_route_deviation_daily) AS min_rdev,
    MAX(avg_route_deviation_daily) AS max_rdev,
    MIN(behavioral_consistency_index_daily) AS min_beh,
    MAX(behavioral_consistency_index_daily) AS max_beh
  FROM fleet_db.analytics.driver_daily
),
norm AS (
  SELECT
    d.driver_id,
    d.day,
    CASE WHEN s.max_anom = s.min_anom THEN 0
         ELSE (d.anomalies_count_daily - s.min_anom) / NULLIF(s.max_anom - s.min_anom,0) END AS anom_n,
    CASE WHEN s.max_ran = s.min_ran THEN 0
         ELSE (d.route_anomalies_daily - s.min_ran) / NULLIF(s.max_ran - s.min_ran,0) END AS ran_n,
    CASE WHEN s.max_geo = s.min_geo THEN 0
```

```
        ELSE (d.geofencing_violations_daily - s.min_geo) / NULLIF(s.max_geo -
s.min_geo,0) END AS geo_n,
    CASE WHEN s.max_rdev = s.min_rdev THEN 0
        ELSE (d.avg_route_deviation_daily - s.min_rdev) / NULLIF(s.max_rdev -
s.min_rdev,0) END AS rdev_n,
    CASE WHEN s.max_beh = s.min_beh THEN 0
        ELSE 1 - ((d.behavioral_consistency_index_daily - s.min_beh) / NULLIF(s.
max_beh - s.min_beh,0)) END AS beh_inv_n
  FROM fleet_db.analytics.driver_daily d CROSS JOIN stats s
)
SELECT
  driver_id,
  day,
  ROUND(
    0.35*beh_inv_n + 0.30*geo_n + 0.20*anom_n + 0.10*ran_n + 0.05*rdev_n
  ,4) AS risk_score_0_1,
 ROUND(
    10 * (0.35*beh_inv_n + 0.30*geo_n + 0.20*anom_n + 0.10*ran_n + 0.05*rde
v_n)
  ,3) AS risk_score_1_10,
FROM norm
ORDER BY risk_score_0_1 DESC



select * from fleet_db.analytics.driver_risk_score

CREATE OR REPLACE TABLE fleet_db.analytics.trip_risk_score AS
WITH base AS (
  SELECT
    trip_id,
    driver_id,
    day,
    CASE WHEN any_anomalous_event THEN 1 ELSE 0 END AS f_anom,
    CASE WHEN any_route_anomaly THEN 1 ELSE 0 END   AS f_route,
    CASE WHEN any_geofencing_violation THEN 1 ELSE 0 END AS f_geo,
    COALESCE(max_route_deviation,0)                  AS route_dev,
    COALESCE(behavioral_consistency_index,0)         AS beh_idx
  FROM fleet_db.analytics.trip_summary
),
-- global mins / maxs for normalization
stats AS (
  SELECT
    MIN(route_dev) AS min_rdev, MAX(route_dev) AS max_rdev,
    MIN(beh_idx)  AS min_beh,  MAX(beh_idx)  AS max_beh
  FROM base
),
norm AS (
  SELECT
    b.*,
    -- flag count and mapped flag score (your chosen mapping: 0,0.33,0.67,1)
    (f_anom + f_route + f_geo) AS flag_count,
    CASE (f_anom + f_route + f_geo)
      WHEN 0 THEN 0.00
      WHEN 1 THEN 0.33
      WHEN 2 THEN 0.67
      ELSE 1.00
    END AS flag_score,
    -- normalized route deviation 0..1
    CASE WHEN s.max_rdev = s.min_rdev THEN 0
         ELSE (b.route_dev - s.min_rdev) / NULLIF(s.max_rdev - s.min_rdev,0) EN
D AS n_route_dev,
    -- normalized inverted behavior 0..1 (higher = worse)
    CASE WHEN s.max_beh = s.min_beh THEN 0
         ELSE 1 - ((b.beh_idx - s.min_beh) / NULLIF(s.max_beh - s.min_beh,0)) E
ND AS n_beh_inv
  FROM base b CROSS JOIN stats s
),
severity AS (
  SELECT
    *,
    -- severity is simple average of the two normalized signals
    (n_route_dev + n_beh_inv) / 2.0 AS severity_raw
  FROM norm
),
risk_scored as (
  SELECT
    s.trip_id,
    s.driver_id,
    s.day,
    s.flag_count,
    ROUND(s.flag_score,2)        AS flag_score,
    ROUND(s.severity_raw,4)      AS severity_0_1,
    -- raw weighted score (already 0..1)
    ROUND(0.7*s.flag_score + 0.3*s.severity_raw,4) AS risk_score_0_1,
    -- 1..10
    ROUND(10 * (0.7*s.flag_score + 0.3*s.severity_raw),3) AS risk_score_1_10,
    RANK() OVER (ORDER BY 0.7*s.flag_score + 0.3*s.severity_raw DESC) AS risk_rank
  FROM severity s
)
select *
from risk_scored
ORDER BY risk_score_0_1 DESC

select * from fleet_db.analytics.trip_risk_score



-- tables to be loaded in colab for ML
select * from fleet_db.analytics.trip_summary

select * from fleet_db.analytics.driver_daily

-- created empty table , the values will be loaded in this table through colab
CREATE OR REPLACE TABLE ANALYTICS.PREDICTIONS (
    TRIP_ID VARCHAR,
    DRIVER_ID VARCHAR,
    ACTUAL INT,
    PRED_PROB FLOAT
);

-- check and verify
select * from fleet_db.analytics.predictions
```