Logistic Operations Database Analysis

1)Database creation

CREATE DATABASE logistics_analytics;

2)Schema

```
-- ==========================
-- LOGISTICS ANALYTICS SCHEMA
-- ==========================

-- DRIVERS
CREATE TABLE drivers (
    driver_id TEXT PRIMARY KEY,
    first_name TEXT,
    last_name TEXT,
    hire_date DATE,
    termination_date DATE,
    license_number TEXT,
    license_state TEXT,
    date_of_birth DATE,
    home_terminal TEXT,
    employment_status TEXT,
    cdl_class TEXT,
    years_experience INT);


-- TRUCKS
CREATE TABLE trucks (
    truck_id TEXT PRIMARY KEY,
    unit_number INT,
    make TEXT,
    model_year INT,
    vin TEXT,
    acquisition_date DATE,
    acquisition_mileage INT,
    fuel_type TEXT,
    tank_capacity_gallons INT,
    status TEXT,
    home_terminal TEXT);
```

```sql
-- TRAILERS
CREATE TABLE trailers (
    trailer_id TEXT PRIMARY KEY,
    trailer_number INT,
    trailer_type TEXT,
    length_feet INT,
    model_year INT,
    vin TEXT,
    acquisition_date DATE,
    status TEXT,
    current_location TEXT);

-- CUSTOMERS
CREATE TABLE customers (
    customer_id TEXT PRIMARY KEY,
    customer_name TEXT,
    customer_type TEXT,
    credit_terms_days INT,
    primary_freight_type TEXT,
    account_status TEXT,
    contract_start_date DATE,
    annual_revenue_potential INT);

-- FACILITIES
CREATE TABLE facilities (
    facility_id TEXT PRIMARY KEY,
    facility_name TEXT,
    facility_type TEXT,
    city TEXT,
    state TEXT,
    latitude DOUBLE precision,
    longitude DOUBLE precision,
    dock_doors INT,
    operating_hours TEXT);

-- ROUTES
CREATE TABLE routes (
    route_id TEXT PRIMARY KEY,
    origin_city TEXT,
    origin_state TEXT,
    destination_city TEXT,
    destination_state TEXT,
    typical_distance_miles INT,
    base_rate_per_mile NUMERIC(12,2),
    fuel_surcharge_rate NUMERIC(12,2),
    typical_transit_days INT);
```

```sql
-- LOADS
CREATE TABLE loads (
    load_id TEXT PRIMARY KEY,
    customer_id TEXT,
    route_id TEXT,
    load_date DATE,
    load_type TEXT,
    weight_lbs INT,
    pieces INT,
    revenue NUMERIC(12,2),
    fuel_surcharge NUMERIC(12,2),
    accessorial_charges INT,
    load_status TEXT,
    booking_type TEXT);

-- TRIPS
CREATE TABLE trips (
    trip_id TEXT PRIMARY KEY,
    load_id TEXT,
    driver_id TEXT,
    truck_id TEXT,
    trailer_id TEXT,
    dispatch_date DATE,
    actual_distance_miles INT,
    actual_duration_hours DOUBLE precision,
    fuel_gallons_used DOUBLE precision,
    average_mpg DOUBLE precision,
    idle_time_hours DOUBLE precision,
    trip_status TEXT);

-- FUEL PURCHASES
CREATE TABLE fuel_purchases (
    fuel_purchase_id TEXT PRIMARY KEY,
    trip_id TEXT,
    truck_id TEXT,
    driver_id TEXT,
    purchase_date TIMESTAMP,
    location_city TEXT,
    location_state TEXT,
    gallons DOUBLE precision,
    price_per_gallon NUMERIC(12,2),
    total_cost NUMERIC(12,2),
    fuel_card_number TEXT);
```

```sql
-- MAINTENANCE RECORDS
CREATE TABLE maintenance_records (
    maintenance_id TEXT PRIMARY KEY,
    truck_id TEXT,
    maintenance_date DATE,
    maintenance_type TEXT,
    odometer_reading INT,
    labor_hours DOUBLE precision,
    labor_cost NUMERIC(12,2),
    parts_cost NUMERIC(12,2),
    total_cost NUMERIC(12,2),
    facility_location TEXT,
    downtime_hours DOUBLE precision,
    service_description TEXT);

-- DELIVERY EVENTS
CREATE TABLE delivery_events (
    event_id TEXT PRIMARY KEY,
    load_id TEXT,
    trip_id TEXT,
    event_type TEXT,
    facility_id TEXT,
    scheduled_datetime TIMESTAMP,
    actual_datetime TIMESTAMP,
    detention_minutes INT,
    on_time_flag BOOLEAN,
    location_city TEXT,
    location_state TEXT);

-- SAFETY INCIDENTS
CREATE TABLE safety_incidents (
    incident_id TEXT PRIMARY KEY,
    trip_id TEXT,
    truck_id TEXT,
    driver_id TEXT,
    incident_date TIMESTAMP,
    incident_type TEXT,
    location_city TEXT,
    location_state TEXT,
    at_fault_flag BOOLEAN,
    injury_flag BOOLEAN,
    vehicle_damage_cost NUMERIC(12,2),
    cargo_damage_cost NUMERIC(12,2),
    claim_amount NUMERIC(12,2),
    preventable_flag BOOLEAN,
    description TEXT);
```

```sql
-- DRIVER_MONTHLY_METRICS
CREATE TABLE driver_monthly_metrics (
     driver_id TEXT,
    month DATE,
    trips_completed INT,
    total_miles INT,
    total_revenue NUMERIC(12,2),
    average_mpg DOUBLE precision,
    total_fuel_gallons DOUBLE precision,
    on_time_delivery_rate DOUBLE precision,
    average_idle_hours DOUBLE precision,
    PRIMARY KEY (driver_id, month));

-- TRUCK_UTILIZATION_METRICS
CREATE TABLE truck_utilization_metrics (
     truck_id TEXT,
    month DATE,
    trips_completed INT,
    total_miles INT,
    total_revenue NUMERIC(12,2),
    average_mpg DOUBLE precision,
    maintenance_events INT,
    maintenance_cost NUMERIC(12,2),
    downtime_hours DOUBLE precision,
    utilization_rate DOUBLE precision,
    PRIMARY KEY (truck_id, month));
```

3)Data import

4) Foreign keys

```
ALTER TABLE loads
ADD CONSTRAINT fk_loads_customer
FOREIGN KEY (customer_id)
REFERENCES customers(customer_id);

ALTER TABLE loads
ADD CONSTRAINT fk_loads_route
FOREIGN KEY (route_id)
REFERENCES routes(route_id);



ALTER TABLE trips
ADD CONSTRAINT fk_trips_load
FOREIGN KEY (load_id)
REFERENCES loads(load_id);


ALTER TABLE trips
ADD CONSTRAINT fk_trips_driver
FOREIGN KEY (driver_id)
REFERENCES drivers(driver_id);


ALTER TABLE trips
ADD CONSTRAINT fk_trips_truck
FOREIGN KEY (truck_id)
REFERENCES trucks(truck_id);


ALTER TABLE trips
ADD CONSTRAINT fk_trips_trailer
FOREIGN KEY (trailer_id)
REFERENCES trailers(trailer_id);
```

```sql
ALTER TABLE fuel_purchases
ADD CONSTRAINT fk_fuel_trip
FOREIGN KEY (trip_id)
REFERENCES trips(trip_id);


ALTER TABLE fuel_purchases
ADD CONSTRAINT fk_fuel_driver
FOREIGN KEY (driver_id)
REFERENCES drivers(driver_id);


ALTER TABLE fuel_purchases
ADD CONSTRAINT fk_fuel_truck
FOREIGN KEY (truck_id)
REFERENCES trucks(truck_id);
```

```sql
ALTER TABLE maintenance_records
ADD CONSTRAINT fk_maintenance_truck
FOREIGN KEY (truck_id)
REFERENCES trucks(truck_id);
```

```sql
ALTER TABLE delivery_events
ADD CONSTRAINT fk_delivery_load
FOREIGN KEY (load_id)
REFERENCES loads(load_id);


ALTER TABLE delivery_events
ADD CONSTRAINT fk_delivery_trip
FOREIGN KEY (trip_id)
REFERENCES trips(trip_id);


ALTER TABLE delivery_events
ADD CONSTRAINT fk_delivery_facility
FOREIGN KEY (facility_id)
REFERENCES facilities(facility_id);
```

```sql
ALTER TABLE safety_incidents
ADD CONSTRAINT fk_incident_trip
FOREIGN KEY (trip_id)
REFERENCES trips(trip_id);


ALTER TABLE safety_incidents
ADD CONSTRAINT fk_incident_driver
FOREIGN KEY (driver_id)
REFERENCES drivers(driver_id);


ALTER TABLE safety_incidents
ADD CONSTRAINT fk_incident_truck
FOREIGN KEY (truck_id)
REFERENCES trucks(truck_id);
```

```sql
-- =========================
-- VALIDATION
-- =========================


SELECT
    conname,
    conrelid::regclass AS table_name
FROM pg_constraint
WHERE contype = 'f'
ORDER BY table_name;



SELECT
    conname,
    convalidated
FROM pg_constraint
WHERE contype = 'f'
ORDER BY conname;
```

5) Fact and Dimension Tables

Fact tables:

- loads
- trips
- fuel_purchase
- maintenance_records
- delivery_events
- safety_incidents

Dimension tables:

- drivers
- trucks
- trailers
- routes
- customers
- facilities

Metrics tables (aggregates):

- driver_monthly_metrics
- truck_utilization_metrics

6) Indexes

```sql
-- ==========================
-- LOADS
-- ==========================

CREATE INDEX idx_loads_customer_id ON loads(customer_id);
CREATE INDEX idx_loads_route_id ON loads(route_id);


-- ==========================
-- TRIPS
-- ==========================

CREATE INDEX idx_trips_load_id ON trips(load_id);
CREATE INDEX idx_trips_driver_id ON trips(driver_id);
CREATE INDEX idx_trips_truck_id ON trips(truck_id);
CREATE INDEX idx_trips_trailer_id ON trips(trailer_id);


-- ==========================
-- FUEL PURCHASES
-- ==========================

CREATE INDEX idx_fuel_trip_id ON fuel_purchases(trip_id);
CREATE INDEX idx_fuel_driver_id ON fuel_purchases(driver_id);
CREATE INDEX idx_fuel_truck_id ON fuel_purchases(truck_id);


-- ==========================
-- MAINTENANCE
-- ==========================

CREATE INDEX idx_maintenance_truck_id ON
maintenance_records(truck_id);


-- ==========================
-- DELIVERY EVENTS
-- ==========================

CREATE INDEX idx_delivery_load_id ON delivery_events(load_id);
CREATE INDEX idx_delivery_trip_id ON delivery_events(trip_id);
CREATE INDEX idx_delivery_facility_id ON
delivery_events(facility_id);
```

```sql
-- ==========================
-- SAFETY INCIDENTS
-- ==========================

CREATE INDEX idx_incident_trip_id ON
safety_incidents(trip_id);
CREATE INDEX idx_incident_driver_id ON
safety_incidents(driver_id);
CREATE INDEX idx_incident_truck_id ON
safety_incidents(truck_id);


-- ==========================
-- DATE AND TIME
-- ==========================

CREATE INDEX idx_trips_dispatch_date ON trips(dispatch_date);
CREATE INDEX idx_loads_load_date ON loads(load_date);
CREATE INDEX idx_fuel_purchase_date ON
fuel_purchases(purchase_date);
CREATE INDEX idx_delivery_scheduled_datetime ON
delivery_events(scheduled_datetime);
CREATE INDEX idx_delivery_actual_datetime ON
delivery_events(actual_datetime);


-- ==========================
-- METRICS
-- ==========================

CREATE INDEX idx_driver_month ON
driver_monthly_metrics(month);
CREATE INDEX idx_truck_month ON
truck_utilization_metrics(month);
```

7) Data Quality (orphan records)

```sql
-- ==========================
-- TRIPS WITHOUT LOADS (0)
-- ==========================

SELECT t.trip_id
FROM trips t
LEFT JOIN loads l ON t.load_id = l.load_id
WHERE l.load_id IS NULL;


-- ==========================
-- LOADS WITHOUT DELIVERY EVENTS (0)
-- ==========================

SELECT l.load_id
FROM loads l
LEFT JOIN delivery_events d ON l.load_id = d.load_id
WHERE d.load_id IS NULL;


-- ==========================
-- FUEL PURCHASES WITHOUT TRIPS (0)
-- ==========================

SELECT f.fuel_purchase_id
FROM fuel_purchases f
LEFT JOIN trips t ON f.trip_id = t.trip_id
WHERE t.trip_id IS NULL;


-- ==========================
-- DELIVERY EVENTS WITH NO FACILITIES (0)
-- ==========================

SELECT d.event_id
FROM delivery_events d
LEFT JOIN facilities f ON d.facility_id = f.facility_id
WHERE f.facility_id IS NULL;
```

8) Views

```
-- ==========================
-- VIEW: Revenue per trip, normalized by mile
-- ==========================


CREATE OR REPLACE VIEW vw_trip_revenue AS
SELECT
    t.trip_id,
    t.load_id,
    l.customer_id,
    l.route_id,
    l.revenue + COALESCE(l.fuel_surcharge, 0) +
COALESCE(l.accessorial_charges, 0) AS total_revenue,
    t.actual_distance_miles,
    CASE
        WHEN t.actual_distance_miles > 0
        THEN (l.revenue + COALESCE(l.fuel_surcharge, 0) +
COALESCE(l.accessorial_charges, 0))
             / t.actual_distance_miles
        ELSE NULL
    END AS revenue_per_mile
FROM trips t JOIN loads l ON t.load_id = l.load_id;


-- ==========================
-- VIEW: Costs per trip, normalized by mile
-- ==========================


CREATE OR REPLACE VIEW vw_trip_costs AS
SELECT
    t.trip_id,
    t.actual_distance_miles,
    COALESCE(SUM(fp.total_cost), 0) AS fuel_cost,
    COALESCE(SUM(si.vehicle_damage_cost +
si.cargo_damage_cost), 0) AS incident_cost,
    CASE
        WHEN t.actual_distance_miles > 0
        THEN COALESCE(SUM(fp.total_cost), 0) /
t.actual_distance_miles
        ELSE NULL
    END AS fuel_cost_per_mile
FROM trips t LEFT JOIN fuel_purchases fp ON t.trip_id =
fp.trip_id LEFT JOIN safety_incidents si ON t.trip_id =
si.trip_id
GROUP BY t.trip_id, t.actual_distance_miles;
```

```sql
-- =========================
-- VIEW: Drivers efficiency and productivity
-- =========================

CREATE OR REPLACE VIEW vw_driver_trip_metrics AS
SELECT
    t.trip_id,
    t.driver_id,
    d.first_name,
    d.last_name,
    t.actual_distance_miles,
    t.actual_duration_hours,
    CASE
        WHEN t.actual_duration_hours > 0
        THEN t.actual_distance_miles / t.actual_duration_hours
        ELSE NULL
    END AS miles_per_hour,
    t.average_mpg,
    t.idle_time_hours,
    de.on_time_flag
FROM trips t JOIN drivers d ON t.driver_id = d.driver_id LEFT
JOIN delivery_events de ON t.trip_id = de.trip_id;



-- =========================
-- VIEW: Trucks operating costs
-- =========================

CREATE OR REPLACE VIEW vw_truck_operating_costs AS
SELECT
    t.truck_id,
    tr.make,
    tr.model_year,
    COUNT(DISTINCT t.trip_id) AS trips_count,
    SUM(t.actual_distance_miles) AS total_miles,
    COALESCE(SUM(m.total_cost), 0) AS maintenance_cost,
    COALESCE(SUM(m.downtime_hours), 0) AS downtime_hours,
    CASE
        WHEN SUM(t.actual_distance_miles) > 0
        THEN COALESCE(SUM(m.total_cost), 0) /
SUM(t.actual_distance_miles)
        ELSE NULL
    END AS maintenance_cost_per_mile
FROM trips t JOIN trucks tr ON t.truck_id = tr.truck_id LEFT
JOIN maintenance_records m ON t.truck_id = m.truck_id
GROUP BY t.truck_id, tr.make, tr.model_year;
```

9) Queries

```sql
-- =========================
-- 1. CUSTOMERS WITH HIGHEST TOTAL REVENUE
-- =========================

SELECT c.customer_id, c.customer_name, SUM(r.total_revenue) AS
total_revenue
FROM vw_trip_revenue r JOIN customers c ON r.customer_id =
c.customer_id
GROUP BY c.customer_id, c.customer_name
ORDER BY total_revenue DESC;
```

```sql
-- =========================
-- 2. MOST FREQUENTLY USED ROUTES
-- =========================

SELECT r.route_id, r.origin_city, r.destination_city,
COUNT(l.load_id) AS load_count
FROM loads l JOIN routes r ON l.route_id = r.route_id
GROUP BY r.route_id, r.origin_city, r.destination_city
ORDER BY load_count DESC;
```

```sql
-- =========================
-- 3. AVERAGE REVENUE PER MILE BY ROUTE
-- =========================

SELECT r.route_id, r.origin_city, r.destination_city,
AVG(v.revenue_per_mile) AS avg_revenue_per_mile
FROM vw_trip_revenue v JOIN routes r ON v.route_id = r.route_id
GROUP BY r.route_id, r.origin_city, r.destination_city
ORDER BY avg_revenue_per_mile DESC;
```

```sql
-- =========================
-- 4. DRIVERS WITH HIGHEST ON-TIME DELIVERY RATE
-- =========================

SELECT driver_id, COUNT(*) AS trips_completed,
AVG(on_time_flag::int) AS on_time_rate
FROM vw_driver_trip_metrics
GROUP BY driver_id
HAVING COUNT(*) >= 10
ORDER BY on_time_rate DESC;
```

```sql
-- =========================
-- 5. TRUCKS WITH HIGHEST REVENUE PER MILE
-- =========================

SELECT t.truck_id, SUM(r.total_revenue) /
SUM(t.actual_distance_miles) AS revenue_per_mile
FROM vw_trip_revenue r JOIN trips t ON r.trip_id = t.trip_id
GROUP BY t.truck_id
ORDER BY revenue_per_mile DESC;




-- =========================
-- 6. ROUTES WITH HIGHEST FUEL COST PER MILE
-- =========================

SELECT r.route_id, r.origin_city, r.destination_city,
AVG(c.fuel_cost_per_mile) AS avg_fuel_cost_per_mile
FROM vw_trip_costs c JOIN trips t ON c.trip_id = t.trip_id JOIN
loads l ON t.load_id = l.load_id JOIN routes r ON l.route_id =
r.route_id
GROUP BY r.route_id, r.origin_city, r.destination_city
ORDER BY avg_fuel_cost_per_mile DESC;




-- =========================
-- 7. MOST PROFITABLE ROUTES PER MILE
-- =========================

SELECT r.route_id, r.origin_city, r.destination_city,
AVG(v.revenue_per_mile - c.fuel_cost_per_mile) AS profit_per_mile
FROM vw_trip_revenue v JOIN vw_trip_costs c ON v.trip_id = c.trip_id
JOIN routes r ON v.route_id = r.route_id
GROUP BY r.route_id, r.origin_city, r.destination_city
ORDER BY profit_per_mile DESC;




-- =========================
-- 8. CUSTOMER PROFITABILITY ANALYSIS
-- =========================

SELECT c.customer_id, c.customer_name, SUM(v.total_revenue -
cst.fuel_cost - cst.incident_cost) AS total_profit
FROM vw_trip_revenue v JOIN vw_trip_costs cst ON v.trip_id =
cst.trip_id JOIN customers c ON v.customer_id = c.customer_id
GROUP BY c.customer_id, c.customer_name
ORDER BY total_profit DESC;
```

```sql
-- =========================
-- 9. PERCENTAGE OF TRIPS OPERATING AT A LOSS
-- =========================

SELECT COUNT(*) FILTER (WHERE (v.total_revenue - c.fuel_cost -
c.incident_cost) < 0)::decimal / COUNT(*) * 100 AS
loss_trip_percentage
FROM vw_trip_revenue v JOIN vw_trip_costs c ON v.trip_id =
c.trip_id;




-- =========================
-- 10. MONTH-OVER-MONTH REVENUE PER MILE TREND
-- =========================

SELECT DATE_TRUNC('month', t.dispatch_date) AS month,
AVG(v.revenue_per_mile) AS avg_revenue_per_mile
FROM vw_trip_revenue v JOIN trips t ON v.trip_id = t.trip_id
GROUP BY month
ORDER BY month;




-- =========================
-- 11. DRIVER PERFORMANCE TREND OVER TIME
-- =========================

SELECT driver_id, DATE_TRUNC('month', dispatch_date) AS month,
AVG(actual_distance_miles / NULLIF(actual_duration_hours, 0)) AS
avg_miles_per_hour
FROM trips
GROUP BY driver_id, month
ORDER BY driver_id, month;




-- =========================
-- 12. DRIVERS / TRUCKS WITH HIGHEST INCIDENT RATE (PER 10000 MILES)
-- =========================

SELECT t.driver_id, COUNT(si.incident_id) * 10000.0 /
SUM(t.actual_distance_miles) AS incidents_per_10000_miles
FROM trips t LEFT JOIN safety_incidents si ON t.trip_id = si.trip_id
GROUP BY t.driver_id
HAVING SUM(t.actual_distance_miles) > 0
ORDER BY incidents_per_10000_miles DESC;
```

```sql
-- ==========================
-- 13. ARE HIGH-REVENUE ROUTES ALSO HIGH-RISK?
-- ==========================

SELECT r.route_id, AVG(v.revenue_per_mile) AS avg_revenue_per_mile,
COUNT(si.incident_id) * 10000.0 / SUM(t.actual_distance_miles) AS
incidents_per_10000_miles
FROM trips t JOIN vw_trip_revenue v ON t.trip_id = v.trip_id JOIN
routes r ON v.route_id = r.route_id LEFT JOIN safety_incidents si ON
t.trip_id = si.trip_id
GROUP BY r.route_id
ORDER BY avg_revenue_per_mile DESC;
```