MetroCycle Analytics: End-to-End Project Documentation

1. Executive Summary

- **Project Name:** MetroCycle Analytics
- **Domain:** Transportation & Logistics
- **Tech Stack:** Python (Data Generation), PostgreSQL (Data Warehousing), Power BI (Business Intelligence)
- **Business Goal:** To solve the "Rebalancing Problem" in bike-sharing systems—preventing stations from becoming empty (lost revenue) or full (user frustration) during rush hours, and identifying high-value routes for infrastructure expansion.

2. Phase 1: Data Simulation (Python)

Since real-world public datasets often lack specific "rebalancing" signals, we generated a custom dataset using the generate_bike_data.py script.

2.1. The Data Logic

- **Scope:** 50 Stations, 800 Bikes, ~150,000 Trips (Jan 1, 2023 – Dec 31, 2023).
- **Geography:** 5 Districts (Financial District, Uptown, Waterfront, University, Suburbs).
- **Patterns:**
    - **Commuter Flow:** Heavy traffic from *Suburbs* to *Financial District* in the AM (7-9 AM) and reverse in the PM (5-7 PM).
    - **Weather Impact:** Ridership decreases by 30-60% during Rain, Snow, or extreme temperatures.
    - **User Types:** 'Subscribers' (Commuters) vs 'Customers' (Tourists/Casual).

2.2. Generated Artifacts

1. **stations.csv**: Contains station_id, capacity, latitude, longitude, and district.
2. **weather.csv**: Daily weather logs with ridership_factor (0.4 for rain, 1.0 for clear).
3. **trips.csv**: Transactional data linked to stations and weather.

3. Phase 2: Database Architecture (PostgreSQL)

The database serves as the transformation engine, turning raw logs into actionable metrics like "Rebalancing Risk."

3.1. Schema Design

We utilize a classic schema with two Dimension tables and one Fact table.

- **stations (Dimension):** Static details about dock locations.
- **weather (Dimension):** Environmental context.
- **trips (Fact):** The core transaction table containing start_time, end_time, start_station_id, etc.

3.2. Critical ETL Steps (Windows Specific)

To avoid "Permission Denied" errors common on Windows PostgreSQL installations:

1. Create a folder C:\Temp.
2. Move generated CSVs there.
3. Use the COPY command with forward slashes:
   COPY trips FROM 'C:/Temp/trips.csv' DELIMITER ',' CSV HEADER;

3.3. Analytical Layers (SQL Views)

Instead of raw queries, we built Views to pre-calculate complex logic for Power BI.

- **v_hourly_station_flow**:
  - **Purpose:** Calculates the "Net Flow" (Inbound - Outbound) per hour.
  - **Logic:** Uses UNION ALL to combine starts (negatives) and ends (positives).
- **v_rebalancing_alerts (The Core Logic)**:
  - **Purpose:** Flags stations at risk during AM Rush Hour (7-9 AM).
  - **Logic:**
    - *High Risk: Empty* = If AM Outflows > (Current Inventory + Inflows).
    - *High Risk: Full* = If AM Inflows > 90% Capacity.
- **v_popular_routes**:
  - **Purpose:** Identifies top O-D (Origin-Destination) pairs.
  - **Use Case:** City planners use this to decide where to build protected bike lanes.

4. Phase 3: Business Intelligence (Power BI)

The dashboard visualizes the SQL insights.

4.1. Data Modeling (Star Schema)

- **Active Relationship:** trips[start_station_id] $\rightarrow$ stations[station_id].
- **Inactive Relationship:** trips[end_station_id] $\rightarrow$ stations[station_id] (Used for "Returns" analysis).
- **Date Relationship:** trips[start_time] $\rightarrow$ weather[date].

4.2. DAX Measures Library

These measures were created in the _Key Measures table:

Measure	DAX Formula	Purpose

Total Trips	COUNTROWS(trips)	Volume analysis.
Utilization Rate	DIVIDE([Total Trips], COUNT(stations[station_id]) * MAX(stations[capacity]) * 365, 0)	Efficiency metric.
Weekend Trips	CALCULATE([Total Trips], WEEKDAY(trips[start_time], 2) > 5)	Segmentation.
Weekday Trips	CALCULATE([Total Trips], WEEKDAY(trips[start_time], 2) <= 5)	Segmentation.
Commuter Score	DIVIDE([Weekday Trips], [Weekend Trips], 0)	High score = Work station.
Net Flow	SUM(v_hourly_station_flow[net_flow])	Rebalancing direction.

4.3. Dashboard Pages & NLQ Prompts

We utilized Power BI's **Q&A** (Natural Language Query) to generate visuals instantly.

Page 1: Operations Commander (Rebalancing)

Target: Fleet Managers identifying where to move bikes *tonight*.

- **Map:** Stations colored by Risk.
  - *Prompt:* "show stations by capacity and am_rush_status as map"
- **Bar Chart:** District-level flow.
  - *Prompt:* "total net flow by district as bar chart"
- **Slicer:** Filter by Weather[Condition] to see how rain affects operations.

Page 2: Strategic Planning

Target: City Planners deciding on infrastructure.

- **Scatter Plot:** Identifying "Cash Cow" vs. "Dead Zone" stations.
  - *Prompt:* "scatter plot of commuter score vs utilization rate by district"
- **KPI Cards:** High-level health check.
  - *Prompt:* "average commuter score as card"
  - *Prompt:* "total trips as card"
  - *Prompt:* "average utilization rate as card"

Page 3: Time Series Analysis

Target: Demand Forecasting.

- **Line Chart:** Hourly demand patterns.
  - *Prompt:* "total trips by hour of day and user type as line chart"

5. Implementation Roadmap (How to Build)

1. **Generate Data:** Run generate_bike_data.py.
2. **Prepare Storage:** Move the 3 generated CSV files to C:\Temp.
3. **Build Database:** Open your SQL tool (pgAdmin) and execute bike_share_analysis.sql.
4. **Connect Power BI:**
   - Get Data $\rightarrow$ PostgreSQL $\rightarrow$ DirectQuery.
   - Select tables and Views.
5. **Model:** Link Tables in the "Model View" tab.
6. **Analyze:** Create the _Key Measures table and paste the DAX formulas.
7. **Visualize:** Use the Q&A Prompts to build the 3 dashboard pages.