Park Usage Analytics Dashboard

LANGARA COLLEGE + CITY OF VANCOUVER

Angeli De Los Reyes, Nay Zaw Lin, Javier Merino, Meyliani Sanjaya

Table of contents

EXECUTIVE SUMMARY

This report outlines the development, deployment, and maintenance of the Park Usage Analytics Dashboard designed to enhance decision-making within the City of Vancouver Park Board. The primary objective was to improve visibility into when and how the public utilize city parks by integrating disparate data sources and implementing predictive modeling techniques. Given the current limitations in real-time park usage monitoring, this project aimed to establish a scalable and interpretable framework to support both operational and strategic planning.

To achieve this, the team designed an interactive Power BI dashboard developed within the Microsoft Fabric environment. This system consolidates multiple data types—including Google Popular Times data, observational counts, weather, park attributes, and holidays & events—into a unified system.

The dashboard is structured to support these core areas of decision-making:

- Smarter Resource Allocation – Optimize the scheduling of park maintenance and staff deployments based on occupancy trends
- Improved Access and Visitor Flow – Identify peak periods to inform crowd management strategies.
- Sustainable and Enjoyable Park Use – Support policies that promote equitable and environmentally responsible use of public spaces.
- Strategic Planning and Investment – Provide data-driven insights to guide long-term infrastructure and amenity development.

CHALLENGES AND DATA CONSTRAINTS

Several challenges shaped the design of the system, including:

- Limited Observational Data – Fewer than 300 observational records were available, with minimal coverage during weekends.
- Lack of Direct User Counts – Data sources such as Google Popular Times do not provide absolute visitor counts. Instead, they present relative busyness scores, which are normalized values indicating how crowded a location is compared to its historical baseline at the same time and day of the week.
- Technological Barriers – Alternative solutions (e.g., sensor networks, camera systems) were not feasible within the current scope due to privacy, cost, and deployment constraints.

To mitigate some of these limitations, the project integrated external contextual data—such as hourly weather conditions, event schedules, and park features—into the

modeling process. This allowed the team to develop predictive tools that estimate hourly/daily park users and forecast hourly occupancy levels over a 7-day horizon.

PROJECT DELIVERABLES

Through the integration of diverse data sources and predictive modeling techniques, the project achieved the following results:

- A Power BI dashboard with three interactive views:
    - Overview – Aggregated occupancy insights across parks and time periods.
    - Park View – Detailed analysis of individual parks, including user estimates and contextual features.
    - Forecast – A forward-looking view displaying hourly occupancy averages and predictions with associated confidence intervals.
- A complete data pipeline built on Microsoft Fabric architecture, powered by Python notebooks, to enable seamless data ingestion, transformation, integration, and analytics.
- A forecasting and estimation pipeline, capable of predicting weekly occupancy trends by park, using historical and live data.

DATA SOURCES

DATA COLLECTED

This project integrates data provided by the Vancouver Park Board along with selected external sources to enrich the analysis and support time series forecasting. The table below defines each dataset and its source.

DATA TYPE	DATA	DEFINITION	SOURCE
Time Series	Google Popular Times (Average/Usual)	Popular times represent the typical park occupancy throughout the day, based on average visitor patterns from the past several months. Each hour between 6 AM and 10 PM is shown relative to the peak usage time of the week, helping visualize when the park is typically more or less busy.	Provided by Park Board
	Google Popular Times (Live)	This data reflects the current level of park activity in real time, relative to typical visitor patterns. It is overlaid on the popular times graph to show whether the park is busier or quieter than usual at that hour.	Provided by Park Board
	Weather (Historical)	Historical weather data for Vancouver	Provided by Park Board
	Weather (Forecasted)	The 7-day forecast dataset from Visualcrossing.com starts one day after the final historical record to ensure continuity in the datetime series.	https://www.visualcrossing.com
Static	Parks Amenities	Count of available park facilities (e.g., washrooms, playgrounds, sports courts), aggregated into a facility score.	Provided by Park Board
	Parks Classifications	A five-tier classification system based on size, service population, and transit access: Local, Community, Neighborhood, Urban Plaza, Destination.	Provided by Park Board

DATA TYPE	DATA	DEFINITION	SOURCE
	Parks ID	Unique Park identifier "ParkID" with associated metadata including description, category, and URL.	Provided by Park Board
	Park Events	Contains a list of scheduled events in each park, including pre-booked, annual, and major events.	Provided by Park Board
	Parks Map	Geospatial polygon data of park boundaries, used for map visualizations (longitude & latitude).	https://opendata.vancouver.ca/explore/dataset/parks-polygon-representation/information/
	BC Holidays	Statutory holiday in British Columbia used to flag unusual day.	https://www2.gov.bc.ca/gov/content/employment-business/employment-standards-advice/employment-standards/statutory-holidays

To ensure consistency and future scalability, especially with external datasets, the following clarifications are provided:

- Weather (Forecasted): Weather data used for forecasting was downloaded from Visualcrossing.com, the same provider used for the historical weather data supplied by the Park Board. This ensures consistency across all weather-related features. The forecast data was downloaded in .csv format and uploaded to the Lakehouse environment. Currently, this dataset is updated manually; however, future implementation may involve automated ingestion through an API, with Azure Realtime Data Services being a potential solution.
- Parks Map: To enable spatial visualization, park polygons (coordinates) were retrieved externally from the City of Vancouver Open Data portal.
- BC Holidays: Statutory holiday dates in British Columbia were obtained from the BC Government website and converted into a structured .csv file. This dataset is used to flag holidays in the time series model.

DATA INGESTION

- **File Formats & Upload Process**

  The team was provided with datasets in CSV and Excel formats, covering a defined period from 30-Apr-2024 to 15-May-2025. In the future, it is anticipated that this file-based ingestion will be replaced by a direct connection to the Park

Board's internal SQL database for more efficient data integration and automation.

- **Folder Structure:**

  The ingestion process follows a structured folder hierarchy, where all raw data files are stored in a designated "raw" directory. The raw data is organized into four subfolders based on data type, as outlined below:

  1. raw/events: Contains the provided events dataset, which has been cleaned and stored in .xlsx format. The file includes the following columns: park_name, start_time, end_time, and event_name.
  2. raw/holidays: Stores a list of statutory holidays in British Columbia in .csv format, with dates formatted as DD/MM/YYYY.
  3. raw/static: Contains non-temporal (static) datasets, including parksAmenities, parksClassification, parksID, and parksMap. Each subfolder contains one or more .csv files.
  4. raw/time_series: Contains time-dependent datasets, including parksLiveTimes, parksObservationalStudy, parksObservationalStudyID, parksUsualTimes, weatherForecast, and weatherVancouver. Each subfolder contains one or more .csv files.

  This structure supports the Lakehouse Medallion Architecture approach and enables clear separation between raw, staged, and transformed data layers.

- **Data Refresh Frequency:**

  Time-series data sources, such as Google Popular Times and weather data, are expected to be updated daily.

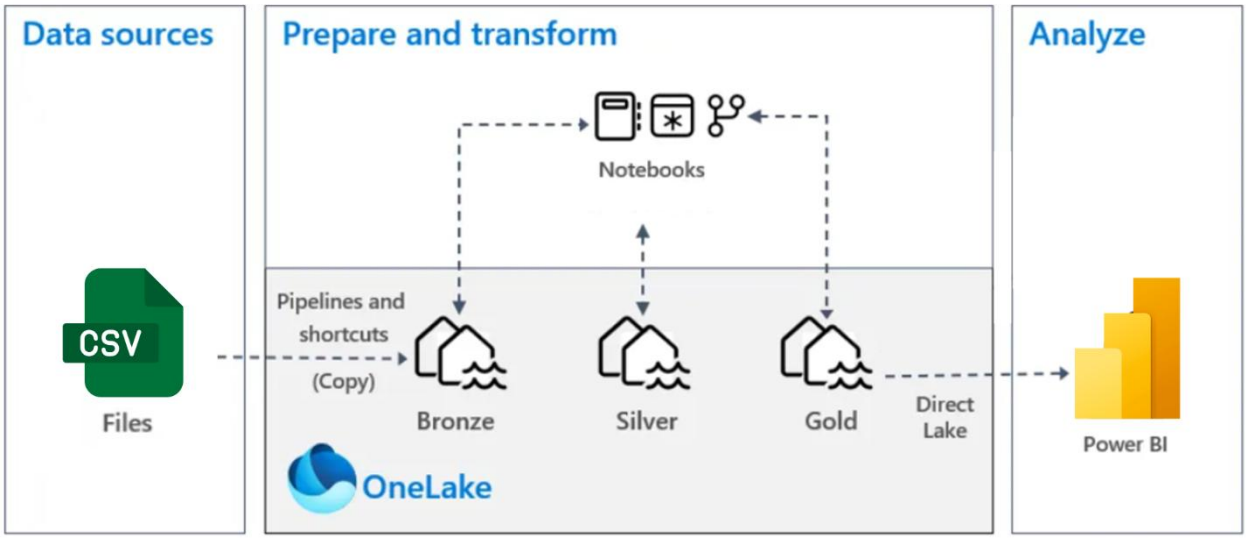
  Static datasets—such as park amenities, classifications, and attributes—are considered slow-changing dimensions and will be refreshed on an as-needed basis.

LAKEHOUSE MEDALLION ARCHITECTURE

The data architecture for this project follows the Medallion Architecture pattern within Microsoft Fabric, which enables scalable, modular, and analytics-ready workflows across the entire data lifecycle. The architecture components collectively support automated ingestion, transformation, prediction, and visualization.

DATA ARCHITECTURE OVERVIEW

The following diagram illustrates the full pipeline from raw data to analytics:



Data Sources

Data is sourced from a combination of files provided by the Park Board and publicly available datasets. These sources include:

- Time Series Data:
  - Google Popular Times (Live and Average/Usual)
  - Historical and Forecasted Weather Data
  - Observational User Counts
- Static Data:
  - Park Metadata (IDs, Classifications, Amenities, Locations)
  - Events Schedule and BC Holidays
  - Park location coordinates from the City of Vancouver Open Data Portal

Initially ingested in CSV and Excel formats, this data is expected to be migrated to a direct SQL connection in future iterations.

⚠️ Important Note: If you need to replace or modify a raw table in the Lakehouse (e.g., by changing column order, dropping fields, or altering types), follow these steps to avoid pipeline or query failures:

1. Drop the existing table from the Fabric Lakehouse to avoid schema mismatch errors.
2. Ensure that the replacement file matches the expected schema exactly, including column names, order, and data types.
3. After loading the new data, refresh the table in the Semantic Model so changes are reflected in downstream reports and measures.
4. Revisit the associated Fabric notebook(s) to verify that the preprocessing logic (column selection, joins, or transformations) still aligns with the updated schema.

Prepare and Transform (Medallion Layers)

- Bronze Layer (Raw Data)

  This layer stores the raw, unprocessed files as-is from external and internal sources. These files maintain source fidelity and provide a historical reference for audits or reprocessing.

- Silver Layer (Staged and Enriched)

  In the Silver layer, the data undergoes standardization, cleansing, deduplication, and schema alignment. Key joins and merges occur here, such as linking park ID, location, classification, and amenity score to a single park attributes table, and combining historical and future weather data into a single weather table.

- Gold Layer (Business-Ready)

  The Gold layer contains refined and structured data, modeled into star schema format with clearly defined dimension and fact tables. It also includes forecast outputs such as user estimates, 7-day hourly forecast graphs and model summaries. This layer is optimized for analytics and directly feeds into Power BI using Direct Lake for high-performance querying.

All Delta tables across the Bronze, Silver, and Gold layers are generated using Microsoft Fabric Notebooks powered by Python (using pandas and PySpark). Data is overwritten with each pipeline run, and versioning is not maintained as historical tracking is not required for this implementation.

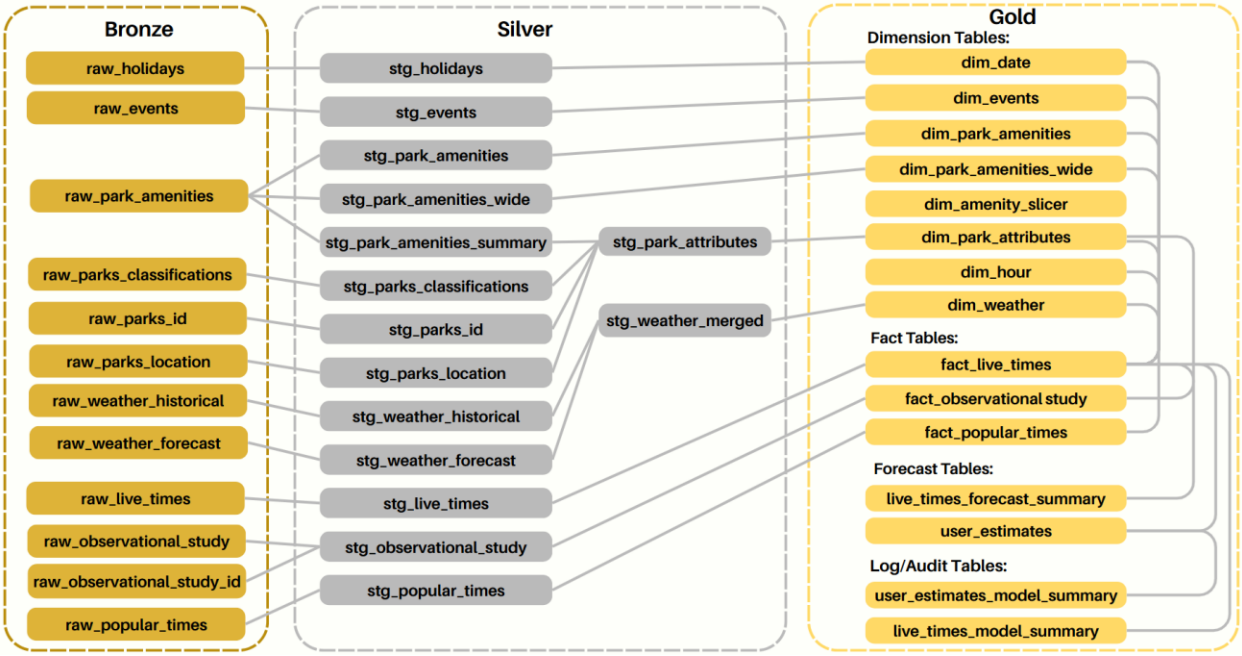
Analyze (Visualization and Reporting)

The Gold layer is connected to a Power BI semantic model, which supports three interactive dashboard views:

- Overview – For aggregated insights across park
- Park View – For detailed park-level analysis
- Forecast – For hourly occupancy averages per month & 7-day occupancy prediction

The dashboards are designed to support operational, planning, and strategic decision-making by the City of Vancouver Park Board, with flexible filtering by park, amenities, classification, events, and more.

DATA TRANSFORMATION: FROM RAW TO GOLD



The detailed data flow diagram above outlines how datasets evolve as they pass through each stage of the pipeline:

- **Bronze Layer (Raw Inputs)**
  - Contains unprocessed tables such as:
    - Time series data such as raw_live_times, raw_popular_times, raw_observational_study, raw_weather_historical, raw_weather_forecast
    - Static attributes such as raw_parks_id, raw_parks_location, raw_park_amenities, and raw_parks_classifications, raw_holidays, raw_events

  - Data ingestion is handled by the "*ingest_01_load_raw*" Fabric notebook, which contains the file paths for all source files. To update the data source locations in the future, modify the file paths directly within this notebook.

- **Silver Layer (Staging and Enrichment)**
  - Key transformations include:
    - Cleansing and restructuring of amenities data (stg_park_amenities, stg_park_amenities_wide, stg_park_amenities_summary)
    - Merging historical and forecasted weather data (stg_weather_merged)
    - Creating wide-format observational data for modeling (stg_observational_study)
    - Integrating park attributes (stg_park_attributes) from stg_parks_id, stg_parks_location, stg_park_amenities_summary, and stg_parks_classifications
  - Use the following Fabric notebooks to modify the pre-processing steps:
    - "*ingest_03_load_stg_2*" – Handles stg_live_times, stg_weather_historical, stg_weather_forecast, stg_weather_merged
    - "*ingest_02_load_stg_1*" – Handles the remaining staging (stg_) tables

- **Gold Layer (Final Output)**

  Final outputs are organized into:

  - **Dimension Tables:** (Modified using the "*ingest_04_load_dim*" Fabric notebook)
    - dim_date – A pre-filled calendar table covering the date range from 2024 to 2030 (*Note:* The code must be updated if the date range needs to be extended in the future). Key columns:
      - DateKey: A unique hash key derived from the date. Used to join with fact tables.
      - IsWeekend: Binary flag (1 = weekend, 0 = weekday). Used in forecasting models.
      - IsHoliday: Binary flag (1 = holiday based on stg_holidays, 0 = not). Used in forecasting models.
      - CurrentInd: Flag used as a visual filter in Power BI (1 = today or earlier; 0 = future dates). Ensures only active dates appear on report pages.
    - dim_hour – A pre-filled table representing each hour from 12:00 AM to 11:00 PM. Key columns:

        - HourKey: A unique hash key derived from the hour. Used to connect with fact tables and support hourly granularity in Power BI visuals.
    - dim_events – A dimension table containing all scheduled park events.
        - EventKey: A unique hash generated from ParkName, StartTime, and EndTime. Used to associate events with park-day combinations in fact tables.
    - dim_park_amenities – A long-format table listing individual amenities by park. This table is used in the Park View page to display all amenities available in the selected park.
        - ParkKey: Foreign key used to join with dim_park_attributes.
        - AmenityKey: Hash derived from ParkName and Amenity (currently not in use).
    - dim_park_amenities_wide – A wide-format version of amenities data. Supports the calculation of KPIs such as Peak/Off-Peak Usage and Live vs. Popular Times % Difference in the Overview page.
        - AmenityKey: Hash key derived from ParkName and PlaceID. Used to join with fact tables for park-level analysis by amenity.
    - dim_amenity_slicer – A simple lookup table containing unique amenity names. Used as a slicer in the Overview page to filter parks by selected amenities.
    - dim_park_attributes – A master table containing comprehensive park-level metadata, including classification, size, location, and other statistics. Created by merging four staging tables into stg_park_attributes.
        - ParkKey: Unique hash generated from PlaceID. Serves as the primary key for joining with fact and dimension tables.
    - dim_weather – Contains hourly weather details per date, including rainfall and snowfall indicators.
        - WeatherKey: Hash generated from Datetime. Used to join with fact tables by hour and day.
        - IsRaining: Binary indicator used in forecasting models (1 = raining, 0 = not)
        - IsSnowing: Binary indicator used in forecasting models (1 = snowing, 0 = not).

- **Fact Tables:** (Modified using "*ingest_05_load_fact_observational_study*", "*ingest_06_load_fact_live_times*", and "*ingest_07_load_fact_popular_times*" Fabric notebooks)

- fact_observational_study - Contains manually recorded headcounts of park visitors, collected during observational studies at specific parks, dates, and hours. Key details:
  - Displayed in the "Observation Study Entries" table on the Park View page.
  - Supports evaluation and validation of model predictions using real-world data.
- fact_live_times – Stores real-time Live Occupancy Percent of parks by hour and day, based on Google Popular Times (Live) data. Feeds KPIs and visualizations showing current user trends across various time periods. Key details:
  - FactLiveKey: Composite hash of ParkKey, DateKey, and HourKey (not currently used).
  - Contains foreign keys to dimension tables (e.g., weather, date, hour, park). Selected dimension attributes are also duplicated to support faster lookups during model execution.
  - Includes a HasEvent binary flag (1 = event present, 0 = none), sourced from dim_events based on overlapping date and time.
  - If key references are unavailable (e.g., park not recognized), corresponding keys default to -1 (representing "Unknown").
- fact_popular_times - Similar in structure to fact_live_times, this table captures Average/Usual Occupancy Percent for each park-hour based on long-term Google Popular Times trends.
  - Used to compare current occupancy against historical norms.

- **Forecast Tables:**
  - live_times_forecast_summary – Stores metadata related to the 7-day hourly occupancy forecast plots for each park. Used in the Forecast page to display the time series forecast chart.
    - Key columns:
      - PlotURL: The web URL of the generated 7-day forecast image for the selected park. Forecast images are saved under Files > forecast > [latest timestamp] each time the prediction is run.
      - ProcessedDateTime: The timestamp indicating when the forecast was generated.
      - ConfidenceScore: A model-derived score that reflects the prediction confidence.
    - Pre-processed using the "*predict_01_timeseries_forecast_1wk*" Fabric notebook.

        - Model training logic can be modified in "*train_02_train_timeseries_model*".
    - user_estimates – Contains the predicted hourly user count per park, based on the model trained from observational data. Displayed on the Park View page to show estimated users and confidence bands.
        - Key columns:
            - PredictedUsers: Estimated number of users for a given park, date, and hour
            - LowerBound: Calculated as PredictedUsers - MAE (lower estimate)
            - UpperBound: Calculated as PredictedUsers + MAE (upper estimate)
        - The predictions are applied to the imputed version of fact_live_times to ensure no gaps between 6 AM and 10 PM (refer to "*train_01_impute_fact_live_times*" on how the table was imputed).
        - Pre-processed using the "*predict_02_user_estimates*" Fabric notebook.
        - Model training logic can be modified in "*train_03_train_user_estimates_model*".

- **Log/Audit Tables:**
    - user_estimates_model_summary – Logs metadata and evaluation results for the user estimation model trained on observational data. Details captured:
        - Evaluation metrics: MAE, RMSE, and R-squared
        - Best parameters selected through grid search
        - Timestamp of model training
        - Each time the model is retrained, a new entry is appended to this table.
        - The trained model artifact is saved in: Files > models > user_estimates
    - live_times_model_summary - Logs evaluation metrics for the 7-day time series forecasting models trained per park. Details captured:
        - Evaluation metrics: MAE, RMSE, R-squared
        - Parameters used during training
        - Timestamp of each model run
        - Models for each park are stored in: Files > models > live_times

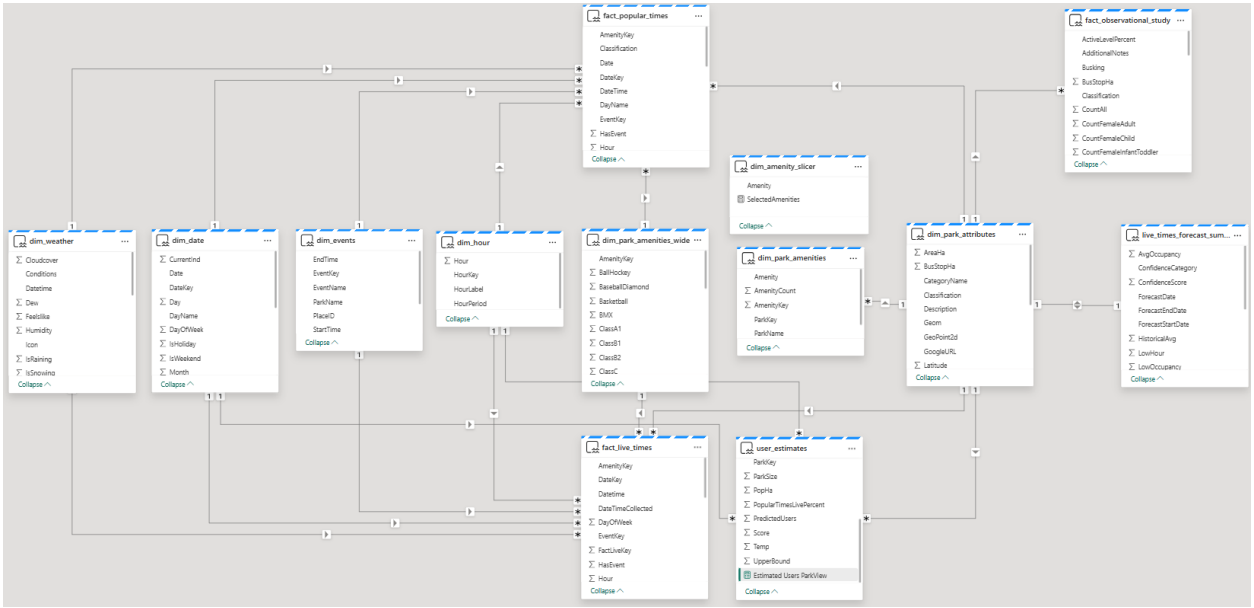
This structured transformation process ensures that all data is not only clean and consistent but also business-ready, supporting descriptive and predictive analytics through a unified semantic model.

SEMANTIC MODEL

To support analytics and visualization in Power BI, a unified semantic model was developed in Microsoft Fabric on top of the Gold layer. This model acts as the foundation for all report pages and enables efficient data slicing, aggregation, and filtering across multiple views.

- **Relationships and Cardinality**

  All relationships follow a one-to-many (1:*) cardinality, where dimension tables filter into fact/forecast tables. Filtering is unidirectional, from dimension to fact. Special handling was applied for time intelligence (via dim_date and dim_hour), amenities (via slicer support) and location-level joins through dim_park_attributes.



  - fact_live_times
    - dim_date → fact_live_times via DateKey (1:*)
    - dim_hour → fact_live_times via HourKey (1:*)
    - dim_park_attributes → fact_live_times via ParkKey (1:*)
    - dim_weather → fact_live_times via WeatherKey (1:*)
    - dim_events → fact_live_times via EventKey (1:*)
    - dim_park_amenities_wide → fact_live_times via AmenityKey (1:*)
    - dim_park_attributes → fact_live_times via ParkKey (1:*)
  - fact_popular_times
    - Same structure and join keys as fact_live_times
  - fact_observational_study
    - dim_park_attributes → fact_observational_study via ParkKey (1:*)
  - user_estimates

        - dim_date → user_estimates via DateKey (1:*)
        - dim_hour → user_estimates via HourKey (1:*)
        - dim_park_attributes → user_estimates via ParkKey (1:*)
    - live_times_forecast_summary
        - Joined to dim_park_attributes via ParkKey (1:1)
    - dim_park_attributes
        - dim_park_amenities → dim_park_attributes via ParkKey (*:1)
    - dim_amenity_slicer
        - Not related to any table; used strictly as a slicer in the dashboard

- **Measures and Calculated Columns**

  A suite of DAX-based calculated measures was developed to power visuals and KPIs across pages. These measures enable dynamic filtering and cross-page consistency:

    - Overview Page:
        - Live vs Popular Times % Diff
        - Parks With Events
        - Filtered Live Off Peak Usage
        - Filtered Live Peak Usage
        - Filtered Live Times Percent (AVG)
        - Filtered Usual Off Peak Usage
        - Filtered Usual Peak Usage
        - Filtered Usual Times Percent (AVG)
        - SelectedAmenities

    - Park View Page:
        - Event on Selected Date
        - Estimated Users ParkView

    - Header:
        - Today Date

  These measures allow slicing by classification, amenity, maintenance area, and time period, providing flexible and scalable analytics.

- **Model Enhancements and Formatting**

  Several enhancements were applied within the Microsoft Fabric Semantic Model to optimize report usability, enable time-based analysis, and ensure accurate sorting and formatting in Power BI:

  1. Date Hierarchy Creation

A custom Year hierarchy was created on the dim_date table to support drill-down functionality across time-based visuals. The hierarchy includes the following levels:

- Year
- MonthName
- DayName

2. Sort Order Configuration

   To ensure correct visual ordering (e.g., chronological rather than alphabetical), specific columns were configured with custom sort columns:

COLUMN	SORT BY COLUMN
HourLabel	Hour
DayName	DayOfWeek
MonthName	Month

   These settings were applied under:
   Semantic Model → Column Properties → Advanced → Sort by column.

3. Data Category Assignment

   To enable correct rendering and interaction within visuals (e.g., clickable links), data types were explicitly categorized where needed. In particular:

COLUMN	DATA CATEGORY
PlotURL	Web URL

This setting allows the forecast image URLs to display as clickable links or embedded images in Power BI. This currently is a Public URL for GitHub due to Lakehouse environment restrictions on hosting public URLs.

Configuration applied via:
Semantic Model → Column Properties → Advanced → Data Category

- **Usage in Power BI Dashboard**

  The Power BI dashboard connects to the semantic model using Direct Lake mode, allowing it to query the latest data directly from the Fabric Lakehouse.

This architecture eliminates the need for scheduled dataset refreshes and enables near real-time performance.

The Power BI report views are powered by the model:

- Overview – Aggregated insights using time series, event, and location filters
- Park View – Detailed visualizations at the park level with amenity and observation overlays
- Forecast – Model-driven predictions with hourly granularity and confidence bands

All visualizations are connected through this model, ensuring consistency across KPIs, slicers, and user interactions. The semantic model also supports Direct Lake mode, offering low-latency performance for end users without the need for scheduled dataset refreshes.

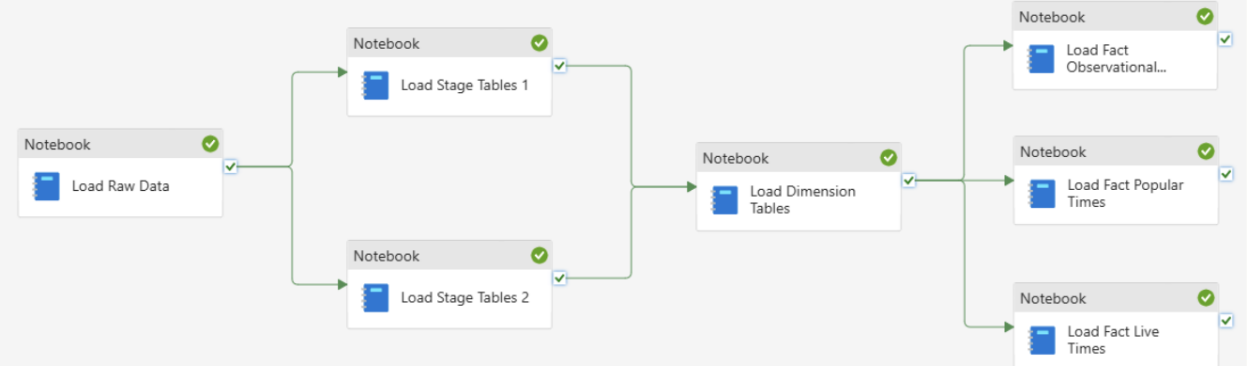
DATA PIPELINES

The table below shows the data flow is orchestrated through four main pipelines

PIPELINE	PURPOSE	SCHEDULE	KEY FEATURES
Data Ingestion + Transformation	Extracts and cleans raw data into dimensions and fact tables	Daily	Full refresh from raw → stage → dims/facts using client-provided CSVs
User Estimation	Predicts daily park usage using XGBoost model	Daily (after ingestion)	Uses 9 features (6 static, 3 weather); outputs daily estimates per park
Train Forecasting Model	Builds a 7-day model per park using historical data	Quarterly or as needed	LightGBM model; uses park events and weather data
Predict Park Occupancy	Produces 7-day hourly forecast per park	Weekly	Includes 7-day historical + 7-day forecast with confidence intervals

1. Data Ingestion + Transformation

The data ingestion and transformation workflow is implemented using a sequence of orchestrated notebooks, designed to process and prepare data in accordance with the Lakehouse Medallion Architecture. The pipeline is fully modular and automated to ensure clean and consistent data delivery across all layers.



- **Notebook: Load Raw Data**
  The pipeline begins with the ingestion of raw .csv files into the Lakehouse Files/raw/ directory. This notebook reads all source datasets—ranging from

static park metadata to time series weather and occupancy data—and performs basic sanity checks (e.g., schema validation, null checks) before landing them into raw tables under the Bronze layer.

- **Notebook: Load Stage Tables 1 & 2**
  Once the raw ingestion is complete, the pipeline splits into two parallel stage-processing notebooks:
  - Stage Tables 1: Handles transformation of datasets such as park amenities, park classifications, and park locations, Google Popular Times, and observational study data
  - Stage Tables 2: Focuses on temporal datasets like weather (historical & forecast) and live times.

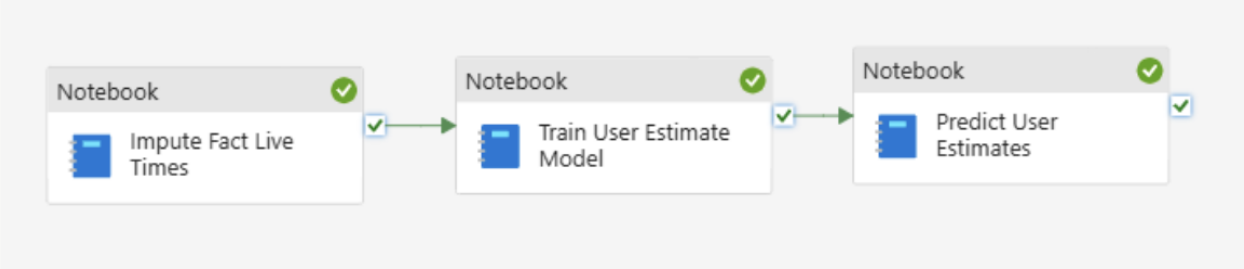
  These notebooks standardize formats, normalize naming conventions, and join related tables to form intermediate structures (e.g., stg_park_amenities, stg_weather_merged, stg_popular_times). These staging tables reside in the Silver layer.

- **Notebook: Load Dimension Tables**
  After both staging notebooks complete successfully, the dimension tables are created. This notebook consolidates multiple stg_ tables to build curated reference dimensions such as:
  - dim_park_attributes (combining classifications, amenities, IDs)
  - dim_weather
  - dim_date and dim_hour for time-based joins
    These dimensions are optimized for consistent joins and downstream analytics in the Gold layer.

- **Notebooks: Load Fact Tables**
  The final stage of the pipeline involves generating fact tables that capture observed or forecasted activity at the park level. These include:
  - Load Fact Observational: Loads field-observed user counts and behaviors for selected parks.
  - Load Fact Popular Times: Integrates Google's historical popularity trends.
  - Load Fact Live Times: Combines live park activity data with timestamp alignment.

These fact tables are used for modeling purposes, except for Fact Live Times, which requires missing value imputation before it can be used in the modeling process.

2. User Estimation



This pipeline runs daily (after the data ingestion pipeline completes) and is designed to predict park-level hourly user counts using a machine learning model. The output is a cleaned and populated user_estimate table containing hourly (and daily when summed) user estimates per park. The pipeline consists of three notebooks executed sequentially:

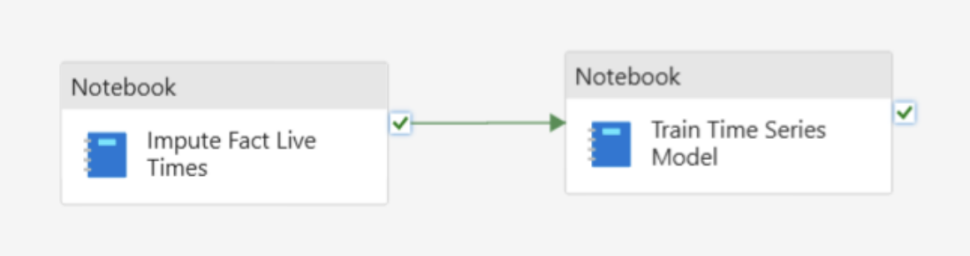
- Notebook: Impute Fact Live Times
  Cleaned and imputed Fact_LiveTimes was used to ensure there are no missing values. Imputation is based on historical patterns to support consistent model input.

- Notebook: Train User Estimate Model
  Builds an XGBoost regression model using 9 features: 6 static attributes (park size, score, classification score, PopularTimesLivePercent, population/hectare, bus stops/hectare) and 3 weather features (temperature, isRaining, isSnowing). This model learns to estimate user counts at a park/hour level.

- Notebook: Predict User Estimates
  Uses the saved trained model to generate hourly & daily user count predictions for all parks. The results are stored in the table for downstream use in forecasting and dashboard reporting.

3. Train Forecasting Model

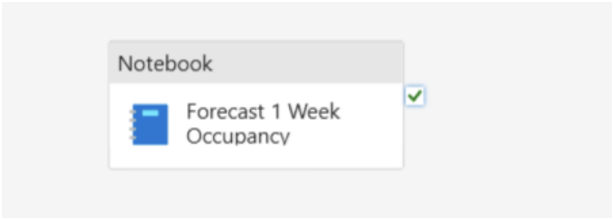
This pipeline is scheduled to run quarterly or as needed and is responsible for building 7-day park-specific occupancy forecasting models. The key output is a set of serialized LightGBM models stored per park, ready for weekly forecast generation in the next pipeline.



- Notebook: Impute Fact Live Times
  Using the same imputed Fact Live Times to ensure consistency and robust analysis, this notebook then passed to Train Time Series Model.

- Notebook: Train Time Series Model
  Trains a separate LightGBM model per park using a combination of engineered lag features and exogenous variables including weather forecasts and known park events. The best model per park is saved in .joblib format to the lakehouse storage for future use in predictions.

3. Predict Park Occupancy



This final pipeline generates 7-day hourly park occupancy forecasts using pre-trained LightGBM models stored in the Lakehouse (Files/models/live_times). It retrieves forecast features (e.g., weather forecast, park metadata) and combines them with the past 7 days of occupancy data to recreate training features like lags and time encodings.

Predictions are generated per park and saved in the live_time_forecast_summary table, with corresponding visualization plots stored under Files/forecast/YYYYMMDD. The folder is dynamically created based on the pipeline's execution date to ensure traceability. The output is then passed to the Power BI dashboard. This pipeline runs on a weekly schedule as the final stage of the automated forecasting system.

POWER BI DASHBOARD FEATURES

KEY FEATURES OVERVIEW

This dashboard provides two core analytics modules:

A. Descriptive Analytics – provides insights through historical park activity analysis.
   - Data Sources: Live occupancy, average busy times, weather, and events
   - Granularity: Hour, day of week, month, year
   - Filters: Park classification, maintenance area, amenities, geolocation
   - KPIs:
     - Peak Hour, Off-Peak Hour
     - Live vs Average Proportion
     - Population per Ha, Bus Stop per Ha

B. Predictive Analytics – provides insights through predicting future occupancy levels (more details can be found in the Predictive Modeling Overview section)
   - Model 1: User Estimation
     - Machine learning model trained on 9 features (6 static, 3 weather)
     - Predicts park-level user counts per day
   - Model 2: Time Series Forecasting
     - Park-specific LightGBM models for 7-day hourly forecasts
     - Incorporates Park events and weather forecasts
   - Granularity: Day of week, operational hour
   - Use Case: Resource allocation and staffing

FILTERS

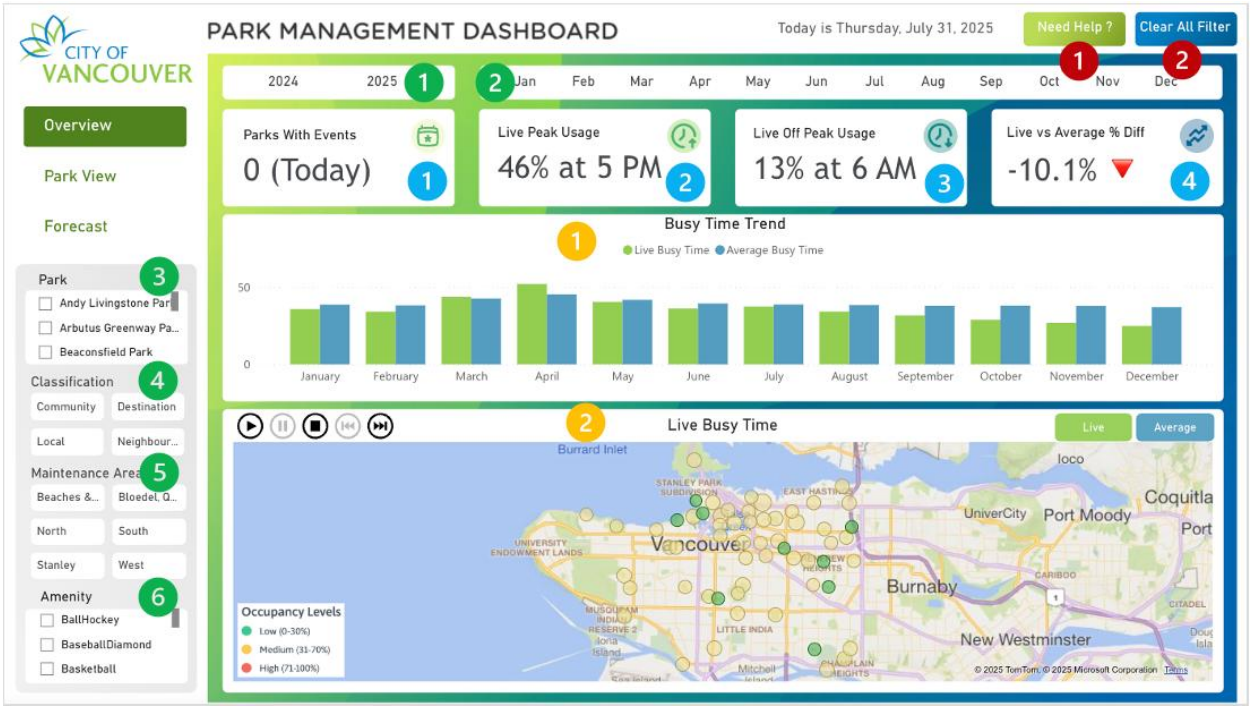
Some back-end filters were configured in the Power BI filter-pane to remove unwanted data and scenarios. The filters applied to the features are as follow:

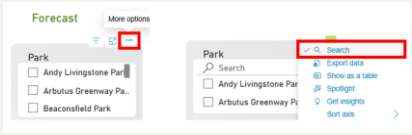
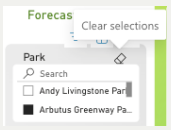
- CurrentInd: display only category 1.
- HourLabel: Display data only from 6:00 to 22:00.
- ParkKey: Display all data that is not -1 or blank. This is to show only the prioritized parks.
- PopularTimesLivePercent: Only displays the data greater than 0 to filter the negative values found in the Live data.

PAGE-LEVEL FUNCTIONALITY & NAVIGATION

A. OVERVIEW

- Purpose:
    a. Provides a city-wide summary of live and average park usage trends with map-based visualizations.
- Key Elements:



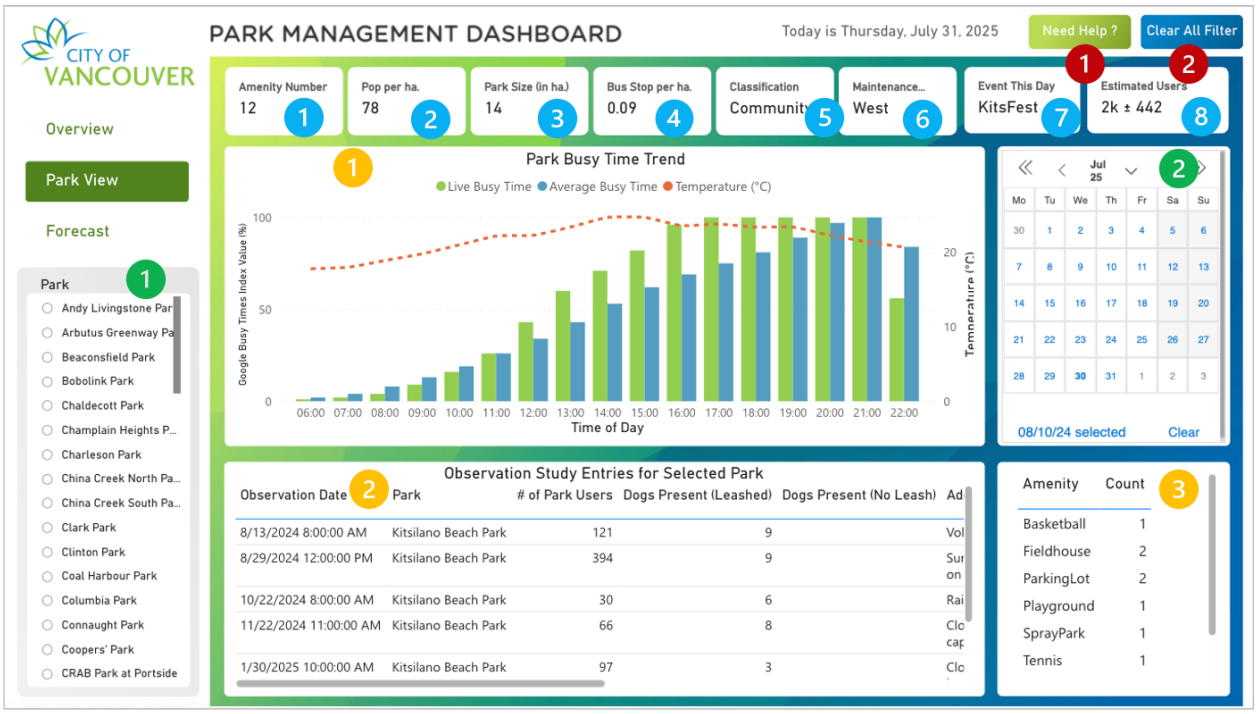
Component	No.	Name	Description
Filters	1	Year	Selects the year to be consider in the analysis Affects all the visuals and KPIs in the page
	2	Month	Selects the year to be consider in the analysis Affects all the visuals and KPIs in the page
	3	Park	Selects a single Park from the list Affects all the visuals, KPIs and filters in the page To enable search, use the ellipsis menu (...) → Search.  Clear the filter by selecting the icon. 

Component	No.	Name	Description
	4	Classification	Selects as single Classification. Affects all the visuals, KPIs and filters in the page.
	5	Maintenance Area	Selects a single Maintenance Area. Affects all the visuals, KPIs and filters in the page.
	6	Amenity	Selects a single Amenity. Affects all the visuals, KPIs and filters in the page. ⚠ If an amenity is selected and the park does not possess such amenity, the visuals do not provide information.
KPI	1	Parks with Events	Displays the unique count of parks with scheduled events during the selected period. If no period is selected, it defaults to showing the count for today. ⚠ The count responds to time, park, classification, and maintenance filters, but not to amenity filters.
	2	Peak Usage	Displays the hour of the day with the highest average occupancy and its corresponding value. It responds to all filters and can switch between Live Peak Usage and Average Peak Usage (Popular times) trough the toggle in the interactive map.
	3	Off Peak Usage	Displays the hour of the day with the lowest average occupancy and its corresponding value. It responds to all filters and can switch between Live Peak Usage and Average Peak Usage (Popular times) through the toggle in the interactive map.
	4	Live vs Average	Displays the proportion of Live vs Average occupancy. Positive = busier than usual; Negative = less busy than usual
Visuals	1	Bar Chart	Displays at all level of granularity the Live and Average busy times. ↑ Moves upwards a level in the hierarchy. ↓ Turns on the Drill down option. When activated, a click in a bar allows to drill down to a lower level in the hierarchy for the selection. ⇊ Allows to go a level down in the hierarchy. ⛙ Expand all down one level in the hierarchy.
	2	Interactive map	The size of the bubbles change based on the occupancy percentage, bigger with higher percentages. Color-coded occupancy (Low 0-30%, Medium 31-70%, High 71-100%). Animation controllers on the upper left corner allow to visualize occupancy hourly. Toggle controller in the upper right corner allow to change between Live and Average Occupancy.

Component	No.	Name	Description
Assistance	1	Help	Provide indications of all the elements in the page.
	2	Clear Filter	Clear all filters in the page.

B. PARK VIEW

- Purpose:
  - Allows deep-dive into an individual park's usage patterns, amenities, and observed user counts.
- Key Elements:

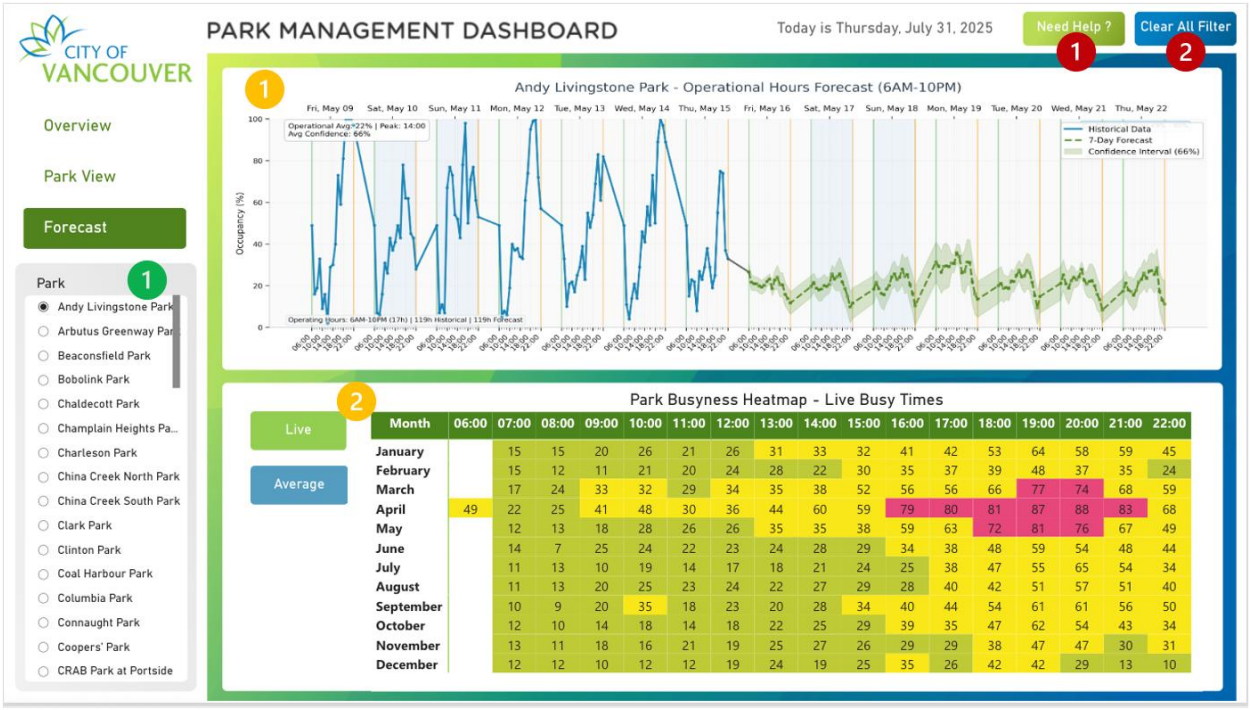


Component	No.	Name	Description
Filters	1	Park	Selects a single Park from the list
	2	Calendar	Display a calendar view to select a single date for analysis. The filter includes controls on the top to navigate through months and years.
KPI	1	Amenity Number	Display the total number of amenities for the chosen park
	2	Population per Hectare	Display an estimate of the population per hectare within a 10-minute walk.
	3	Park Size	Displays the park size in hectares.
	4	Bus Stop per Hectare	Displays an estimate of the number of bus stops per hectare.
	5	Classification	Displays the category of the chosen park.

Component	No.	Name	Description
	6	Maintenance	Displays the maintenance area of the chosen park.
	7	Event this Day	Displays the name of the event celebrated in the park in the chosen date.
	8	Estimated Users	Estimated Users for selected day or hour ⚠ Prediction error (±n users) is based on average model accuracy across all parks. This may be relatively large for parks with low visitor counts.
Visuals	1	Bar Chart	Displays hourly bar chart of Live vs Average occupancy and the temperature of the selected date in degree Celsius.
	2	Observation Study Table	Displays the Observation Study data available for the chosen park and date.
	3	Amenity Table	Displays the list of amenities and its count for the chosen park.
Assistance	1	Help	Provide indications of all the elements in the page.
	2	Clear Filter	Clear all filters in the page.

C. FORECAST VIEW

- Purpose:
  - Provides a vision of the forecast calculated based on the live occupancy data, and a Heath map for evaluating trends.
- Key Elements:



Component	No.	Name	Description
Filters	1	Park	Selects a single Park from the list
Visuals	1	Line Chart	Presents a 7-day hourly occupancy forecast with confidence intervals to support resource planning. The forecast begins with the last available Google Live Times data, covering 1 week of historical data followed by a 7-day projection.
	2	Heat Map	Displays the mean occupancy trends for each month, broken down by hourly intervals within operational hours (6 AM to 10 PM), providing a deeper look into typical park busyness patterns over time. Toggle controller in the upper left corner allows to change between Live and Average Occupancy. ⚠ The toggle affects only the heatmap.
Assistance	1	Help	Provide indications of all the elements in the page.
	2	Clear Filter	Clear all filters in the page.

PREDICTIVE MODELING OVERVIEW

MODEL 1: USER ESTIMATION

The user estimation model predicts the number of park visitors on a specific hour using historical observational data and associated contextual features. The model was designed to augment sparse observational counts by estimating park usage for hours where no direct measurements were available.

Note: The model outputs are not disaggregated by user type (e.g., adults, children) and are only available at the granularity of one park per hour per date. It is not a per-park model; a single XGBoost model was trained using all parks.

Model Design

- **Input Features (9 total)**
  - Static Features:
    - Park Size
    - Classification Score
    - Park Score
    - Popular Times Live Percent
    - Population per Hectare
    - Bus Stops per Hectare
  - Weather Features:
    - Temperature
    - IsRaining (Binary)
    - IsSnowing (Binary)

- **Training and Testing**
  - The model was trained using approximately 300 observational records, of which around 200 had matching Google Live Popular Times percentages at the hourly level and were used for supervised learning.
  - After training, predictions were generated using an imputed version of the Google Live Popular Times data (*imputed_fact_live_times*). This approach ensured hourly coverage between 6 AM and 10 PM, even on days with missing or incomplete live data. The resulting estimates were saved in the *user_estimates* table for integration into the dashboard
  - Target variable: Observed number of park users per hour

- **Performance Comparison**

MODEL TYPE	MAE	RMSE	R^2
Baseline Model (6 features)	37.96	49.63	-0.131
Enhanced Model (6 + 3 weather features)	25.99	50.55	0.249

The table above compares the performance of two models trained to estimate hourly park user counts:

- A Baseline Model using 6 features (static features), and
- An Enhanced Model that includes 3 additional weather-related features (temperature, rainfall, snow).

Interpretation of Evaluation Metrics:

- Mean Absolute Error (MAE): Represents the average magnitude of prediction errors, regardless of direction.
    - The Enhanced Model achieved a lower MAE (25.99) compared to the Baseline (37.96), indicating a more accurate average prediction of user counts across hours and parks.
- Root Mean Squared Error (RMSE): Measures the square root of the average squared errors. It gives higher weight to larger errors.
    - Both models showed similar RMSE values, suggesting comparable performance when considering larger deviations. However, RMSE alone can be sensitive to outliers and should be interpreted alongside MAE.
- R-squared (R^2): Represents the proportion of variance in the observed data explained by the model (ranges from $-\infty$ to 1).
    - The Baseline Model had a negative R^2, indicating it performed worse than a simple average (mean-based) predictor.
    - In contrast, the Enhanced Model achieved a positive R^2 of 0.249, showing meaningful explanatory power—particularly on days influenced by weather conditions.

Although RMSE was comparable across models, the enhanced version produced lower average error (MAE) and greater interpretability (positive R^2), especially on weather-sensitive days. This supports the inclusion of weather variables in future iterations of the model.

Dashboard Integration

The Estimated Users output is integrated into the Park View page of the Power BI dashboard. By default, the card visual displays:

- The total estimated users for the selected park and date
- A ± margin of error, derived from the average model MAE across all parks

If a user selects a specific hour of the day, the card updates to reflect the estimated number of users for that hour, along with the same error margin.

Refer to the *Estimated Users ParkView* measure for the underlying logic.

Model Maintenance and Frequency

The model is retrained and evaluated daily, triggered as part of the Fabric pipeline after the ingestion of new weather data and live occupancy percentages.

Future Improvements

- Data Augmentation: Uploading additional observational records, particularly from underrepresented parks and peak times, would enhance model accuracy.
- Per-Park Models: With sufficient data volume, separate models can be trained for high-traffic parks to better capture local patterns.
- Model Generalization: Further feature engineering (e.g., event flags, time since last rainfall) may improve predictive power.

MODEL 2: TIME SERIES FORECASTING

The time series forecasting leverages a custom-built ModularMultiParkLightGBM class to predict park occupancy (PopularTimesLivePercent) across multiple parks using time series data. The model architecture is modular, allowing separate LightGBM models to be trained per park, with flexible incorporation of exogenous and temporal features.

Note: The model trained with predicting data for 17-hours that reflected from the park operational hours.

Data Preprocessing

To ensure consistent and reliable input for time series forecasting, the fact_live_times dataset was preprocessed with a structured imputation strategy. Since real-world occupancy data (e.g., Google Live Popular Times) often contains missing or invalid values, imputation was critical to maintain hourly continuity, especially for generating lag features and preserving model accuracy.

The process began by loading the data from a Spark table into a Pandas DataFrame, followed by checks for duplicates and missing values. Rows with missing values in essential predictors—such as DateTimeCollected, ParkName, Hour, weather indicators, and calendar flags—were removed. The target variable, PopularTimesLivePercent, had -100 placeholders replaced with NaN to indicate missing data.

A key technique was gap profiling, which quantified the frequency, average length, and maximum span of missing data per park. This profiling informed the selection of one of three imputation methods:

- **Enhanced Groupby Imputation:** Fills missing values using logical groupings like Hour, IsWeekend, or DayOfWeek. Suitable for small gaps and limited data.
- **Temporal Trend Imputation:** Uses linear regression on a time index to capture trends and fill longer or structured gaps.
- **LightGBM-Based Imputation:** Trains a model using contextual features (e.g., Temp, IsHoliday, HasEvent) to predict missing values. Effective for frequent but scattered gaps.

The imputation method was selected dynamically using metrics like TotalMissing, MaxConsecutiveGap, AvgGapLength, and MissingBlocks. This adaptive approach ensured each park's data was clean and complete before being passed into the forecasting model.

- **Input Features (13 total)**
    - Cyclical Time Features:
        - HourSin
        - HourCos
        - HourOp (Hour index adjusted to operational range 6AM–10PM)

    - Lag Features (per park):
        - Lag_1, Lag_2, Lag_3, Lag_17 (hourly lagged values of target variable)

      Note: At 6AM, lag values are imputed using average values by DayOfWeek to address overnight gaps.

- **Weather & Calendar Features:**
    - Temp
    - IsRaining (Binary)
    - IsSnowing (Binary)
    - IsWeekend (Binary)
    - IsHoliday (Binary)
    - HasEvent (Binary)

- **Training and Testing**

  The forecasting models were trained per park using the imputed dataset (imputed_fact_live_times) with complete hourly coverage from 6 AM to 10 PM. A time-based split was used, where the latest 20% of the data was reserved for testing to simulate future predictions. This ensured chronological integrity and prevented data leakage from future to past.

Each park's model was trained on historical lag features, weather indicators, event flags, and calendar context, and two model variants were developed:

- **Baseline models** with default LightGBM settings
- **Optimized models** tuned via Optuna for hyperparameter search

After training, predictions were made on the test set and passed to the evaluation pipeline.

- **Performance Evaluation**

  Model performance was evaluated using standard regression metrics to measure accuracy and generalization:

  - **MAE (Mean Absolute Error)** – Captures the average absolute difference between predictions and actual values.
  - **RMSE (Root Mean Squared Error)** – Penalizes large errors more, emphasizing the impact of outliers.
  - **R^2 (Coefficient of Determination)** – Measures the proportion of variance in the target variable explained by the model.

  For each park, both baseline and optimized models were compared using these metrics. To statistically validate improvements from hyperparameter tuning, paired t-tests were performed across all parks using the MAE results. This provided insight into whether observed improvements were statistically significant or due to chance.

  This evaluation framework ensures the model not only fits the training data well but also performs reliably on future, unseen data—making it suitable for deployment in operational forecasting dashboards. The summary of performance evaluation is stored in a table in the lakehouse.

- **Forecasting & Prediction Pipeline**

  Once the per-park LightGBM models were trained and validated, they were used to generate 1-week ahead forecasts for hourly park occupancy between 6 AM and 10 PM. This prediction pipeline ensures consistent application of trained models on new data, maintaining the same feature logic and integrity established during training.

  - **Forecast Data Construction**

    To prepare input data for inference, the pipeline constructs a complete and enriched forecast dataset using the following steps:

    1. Base Timeline Initialization

The process begins by loading future weather data from the stg_weather_forecast table. To ensure the model only forecasts future conditions, the data is filtered to include only timestamps that occur after the most recent time in the training dataset. Additionally, records are restricted to operational hours between 6 AM and 10 PM to match the temporal scope used during model development.

2. Cross Join with Park Metadata
   To ensure each park receives predictions for every future time slot, the filtered weather data is cross-joined with the list of trained parks. This list is dynamically extracted from saved model filenames, creating a full grid of (Datetime × ParkName) combinations. This step guarantees that all parks are forecasted uniformly across the defined time horizon.
3. Contextual Feature Merging
   To recreate the same feature structure used during training, the following joins are applied:

SOURCE TABLE	MERGED FEATURES
dim_park_attributes	ParkKey, PlaceID
dim_date	DayOfWeek, IsWeekend, IsHoliday
dim_events	HasEvent – computed if Datetime falls within any event window (StartTime to EndTime) for specified park

This step enriches each forecast row with weather, calendar, and event context for accurate modeling.

- **Feature Engineering for Inference**
  Once all joins are complete, the pipeline engineers the same features used in training:
  - Cyclical Time Features: HourOp, HourSin, HourCos
  - Weather Flags: Temp, IsRaining, IsSnowing
  - Calendar Indicators: IsWeekend, IsHoliday
  - Event Context: HasEvent

  Note: Lag features (e.g., Lag_1, Lag_2) are not computed during future inference. In production, these may be handled using recent historical values or fallback logic.

- **Model Inference and Prediction**
  Forecasting is performed separately for each park to ensure predictions align with that park's unique usage patterns and training data. The process involves the following steps:
  - **Load model:** The trained LightGBM model for each park is loaded from the saved .joblib file.

- **Prepare inputs:** All engineered features—time encodings, weather indicators, and event/contextual flags—are aligned to match the structure used during training.
- **Generate predictions:**
  The model then generates predictions for the target variable, PopularTimesLivePercent, on an hour-by-hour basis for the 7-day forecast horizon. These predictions represent estimated occupancy percentages during operational hours and are based entirely on future inputs and the park's historical training patterns.

- **Post-processing and Output**
  The final stage of the prediction pipeline produces two key outputs:

  1. Forecast Summary Table
  All hourly predictions for each park are compiled into a unified table named live_times_forecast_summary.
  - Includes fields such as Datetime, ParkName, and PopularTimesLivePercent.
  - Serves as the main data source for reporting.
  - Stored as a Delta table within the Microsoft Fabric environment.

  2. Forecast Visualization Plots
  Visual plots are generated for each park, illustrating predicted occupancy trends across the 7-day, 17-hour (6 AM–10 PM) forecast window.
  - Plots provide an intuitive view of forecasted usage patterns.
  - Due to limitations in Microsoft Fabric storage, plots are saved externally to GitHub public storage.
  - Each plot is accessible via a unique GitHub URL.

Dashboard Integration

The forecast plots are integrated into the Power BI dashboard by referencing a public GitHub URLs as the storage. Each plot is saved during the prediction pipeline with a consistent naming convention based on park name and forecast date.

In the dashboard, when a user selects a park and date range, the corresponding image is dynamically loaded from GitHub using a parameterized URL. This method allows seamless embedding of visual forecasts without relying on internal Microsoft Fabric storage.

Model Maintenance & Frequency

The training pipeline is recommended to run quarterly, as full model retraining takes about 2 hours and 45 minutes. This schedule ensures the models stay up to date with seasonal patterns while managing computational cost.

The prediction pipeline is designed to run weekly, generating 7-day forecasts. However, it depends on the continuity between the most recent training data and future inputs—especially for lag-based features.

To maintain consistency, the prediction data should directly follow the training period. If training data becomes stale, a lightweight update or retraining may be required.

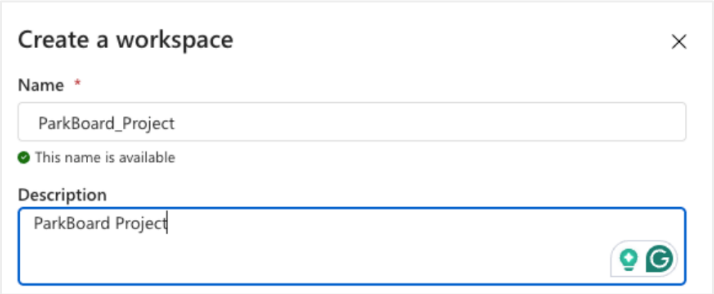
Future Improvements

- Data Quality: Consider improving the accuracy of `PopularTimesLivePercent` by incorporating more reliable occupancy signals (e.g., sensor-based or validated counts), reducing dependence on potentially biased or manipulated live popularity metrics.
- Weather Data: Automate weather data ingestion by connecting a real-time forecast API directly to the lakehouse, removing manual updates.
- Internal Plot Hosting: Move visual forecast plots from GitHub to internal Microsoft Fabric storage or OneLake for better accessibility, and governance.
- End-to-End Automation: Use Fabric's scheduling and pipeline tools to automate quarterly retraining and weekly prediction workflows, including data updates and alerts.

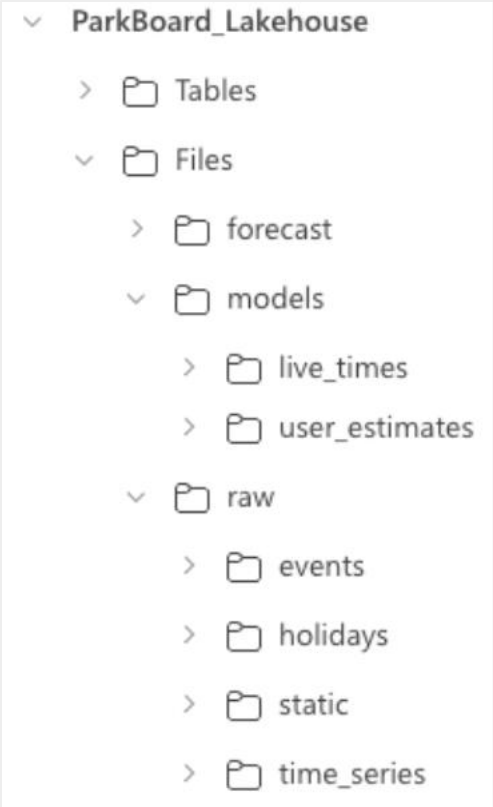
MIGRATION & INSTALLATION

Step-by-step instructions to migrate the following to client's Fabric Workspace:

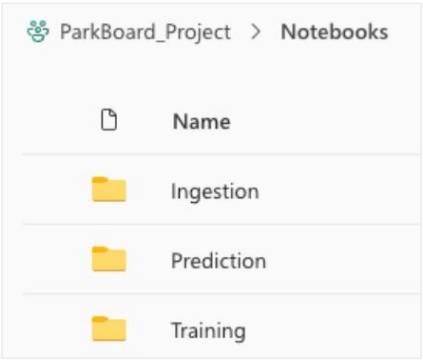
1. Create the new Workspace (ParkBoard_Project) in Fabric



2. Set up the OneLake folder hierarchy and copy the forecast, models, and raw data packages using OneLake File Explorer



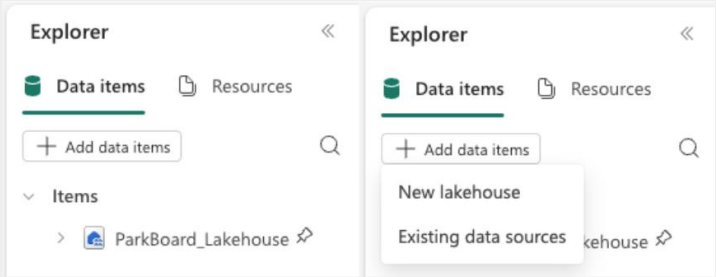
3. Import Fabric Notebooks
    - Create Notebooks folder and upload the Fabric notebooks here.



    - Update *ingest_01_load_raw.ipynb* hardcoded file paths (e.g., abfss://) to match with your workspace.

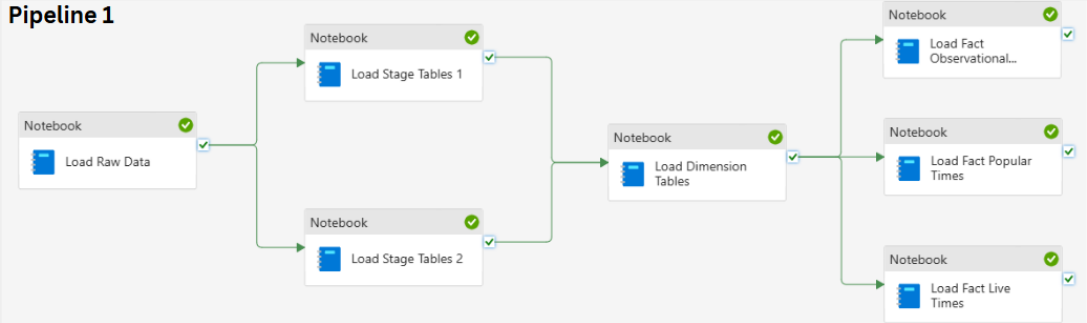
```
# Parks Location (CSV)
spark.read.option("header", True).option("sep", ";") \
    .csv("Files/raw/static/parksMap/parks-polygon-representation.csv") \
    .write.format("delta").mode("overwrite").saveAsTable("raw_parks_location")

# Park amenities (Excel)
amenity_df = pd.read_excel("/lakehouse/default/Files/raw/static/parksAmenities/parkAmenityTable.xlsx", header=0)
amenity_spark_df = spark.createDataFrame(amenity_df)
amenity_spark_df.write.format("delta").mode("overwrite").saveAsTable("raw_park_amenities")
```

- Verify that each notebook is correctly linked to its intended Lakehouse data source. Otherwise, change the correct data source from Existing data sources.



4. Rebuild data pipelines and run the full ingestion-to-prediction flow

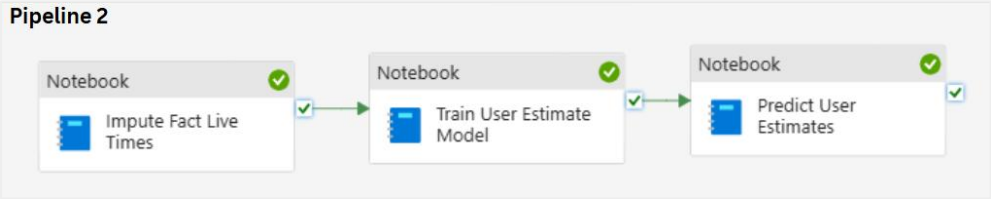
- Pipeline 1: Data Ingestion + Transformation



Notebook Usage in Pipeline 1

PIPELINE STEP	NOTEBOOK NAME
Load Raw Data	ingest_01_load_raw
Load Stage Tables 1	ingest_02_load_stg_1
Load Stage Tables 2	ingest_03_load_stg_2
Load Dimension Tables	ingest_04_load_dim
Load Fact Observational Study	ingest_05_load_fact_observational_study
Load Fact Live Times	ingest_06_load_fact_live_times
Load Fact Popular Times	Ingest_07_load_fact_popular_times

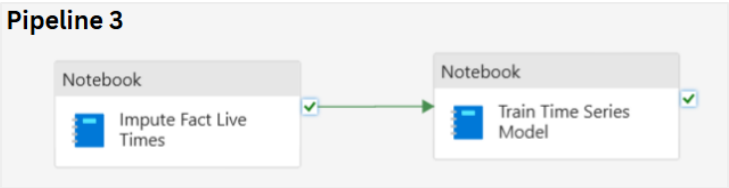
- Pipeline 2: User Estimation



Notebook Usage in Pipeline 2

PIPELINE STEP	NOTEBOOK NAME
Impute Fact Live Times	train01_impute_fact_live_times
Train User Estimate Model	train_03_impute_user_estimates_model
Predict User Estimates	predict_02_user_estimates

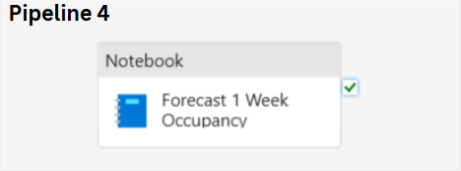
- Pipeline 3: Train Forecasting Model



Notebook Usage in Pipeline 3

PIPELINE STEP	NOTEBOOK NAME
Impute Fact Live Times	train_01_impute_fact_live_times
Train Time Series Model	train_02_train_timeseries_model

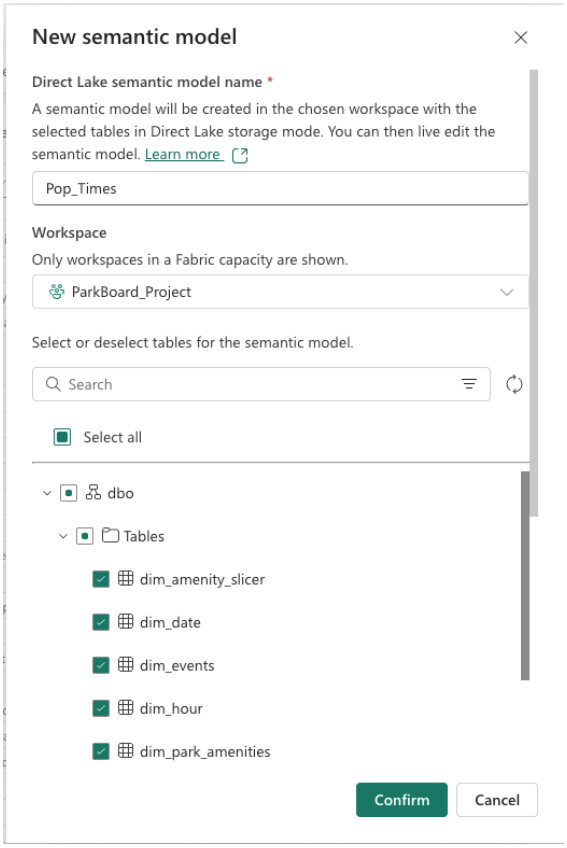
- Pipeline 4: Predict Park Occupancy (1 week)



Notebook Usage in Pipeline 4

PIPELINE STEP	NOTEBOOK NAME
Forecast 1 Week Occupancy	predict_01_timeseries_forecast_1wk

5. Rebuild the Semantic Model
    - Pre-req: Data pipeline must be run to populate tables
    - Due to current limitations in Fabric Web, measures must be recreated through code.
    - Create the new model
        - Go to existing ParkBoard_Lakehouse
        - Click New semantic model from menu



Note: After that, please refer to the Semantic Model section and follow the steps provided, including configuring relationships and cardinality.

6. Setup Power BI Dashboard

- Connect dashboard visuals to the semantic model and publish the Power BI Report (Park Analytics Dashboard) to new Workspace

7. As part of the integration, we used GitHub repository to host the generated forecast plot images from Fabric, allowing us to access and display them directly within the Power BI dashboard.
    - GitHub Steps:
        - Prepare the new repository on existing account (If we don't have the account create the new one at https://github.com)
        - Create the new Organization (Langara-DataHub)
        - Create the new Repository (Van-City-Project) under that organization
        - Create the new branch (develop)
        - Create the new folder (forecast_plots) under develop branch
        - Push generated plot files from Fabric to Github.

Folder Structure

```
develop/
└── forecast_plots/
    └── 20250728/
        └── Andy_Livingstone_Park.png
```

8. Checklist to confirm successful migration

No.	Checklist	Status
1.	Create the new Workspace	
2.	Set up the OneLake folder hierarchy and copy the forecast, models, and raw data packages	
3.	Import Fabric Notebooks	
4.	Rebuild data pipelines and run the full ingestion-to-prediction flow	
5.	Rebuild the Semantic Model	
6.	Setup Power BI Dashboard	
7.	Configure storage and public access URLs for forecast plot display in Power BI	