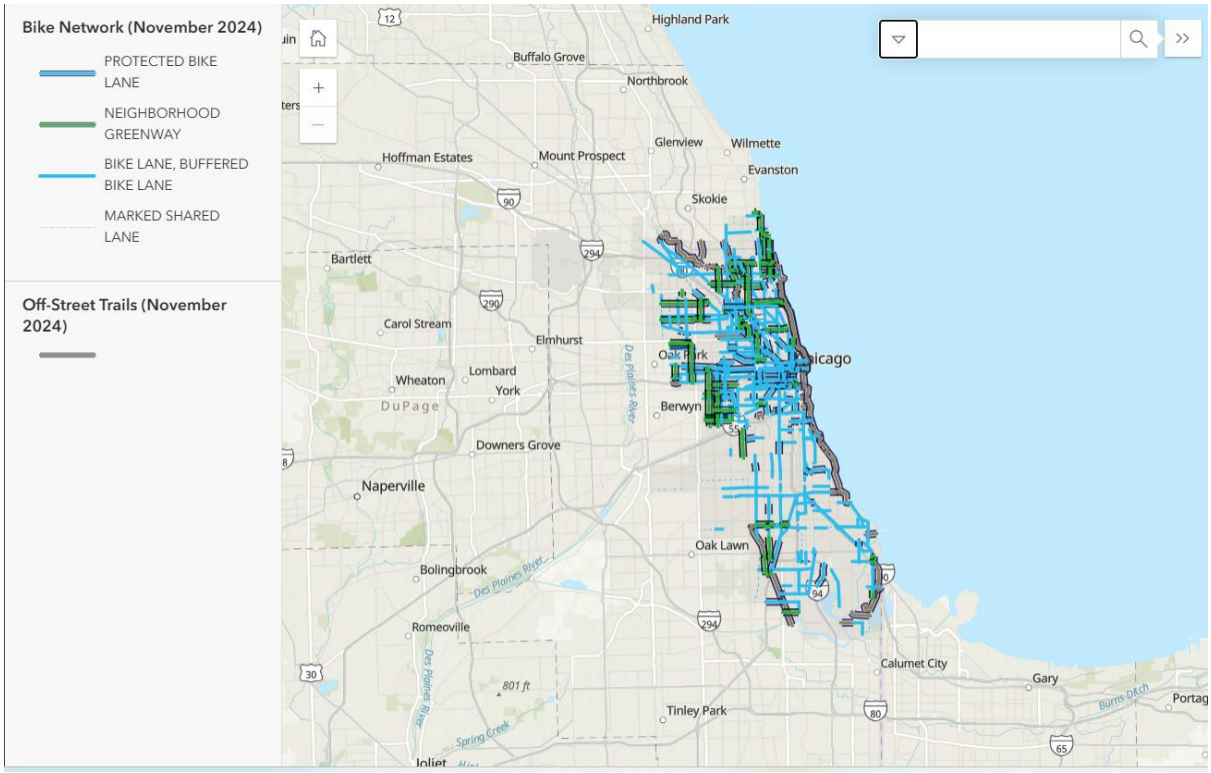

Chicago Bike Sharing Project

A Comprehensive Case Study on Urban Mobility Solutions



Authors:

Ahmed Laarfi

Date:

January 17, 2025

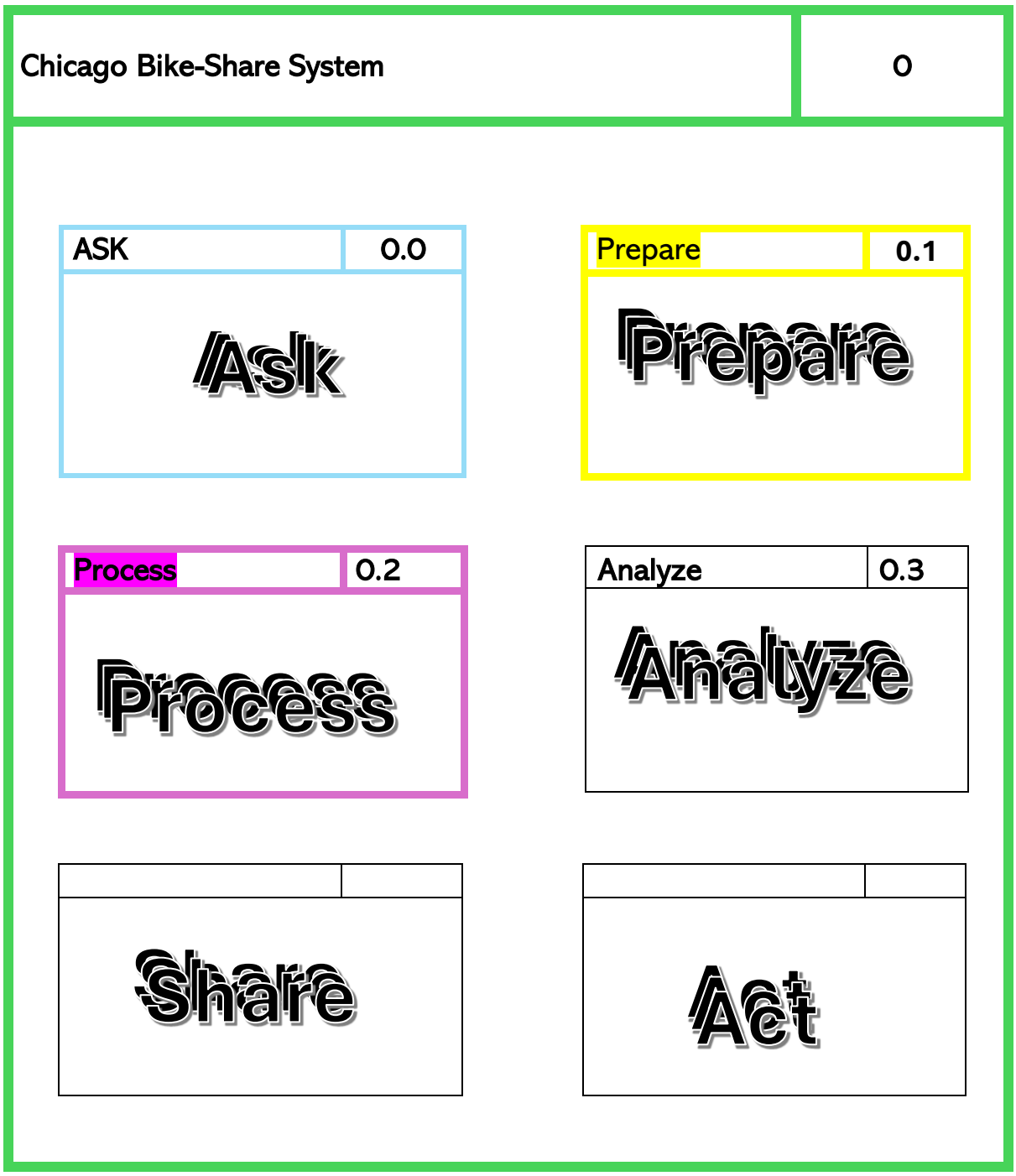
Institution/Organization:

Coursera

Case Study: Chicago Bike-Share Project

Introduction:

The Chicago Bike Sharing Project is designed to promote sustainable transportation, enhance the quality of urban life, and provide economic advantages. By facilitating prompt and secure access to bicycles, the initiative aims to alleviate traffic congestion and encourage healthier lifestyle choices among residents. Nevertheless, the project encounters significant challenges arising from the complexities of capitalism and the influence of powerful industries, such as health insurance and automotive sectors.



0. The whole System Phases of the Project

0.0. Ask Phase [Three Stages]

Ask	0.0

System Definition and Data Types	0.0.0
• Understanding the User Base • Membership Structure • Geographical Boundaries • Station Distribution	

Bicycle Types and Sharing Mechanism	0.0.1
• Bicycle Selection • Ownership and Contribution Programs	

System Manual and Documentation	0.0.2
• Administrative Models • File Management	

Phase 1: System Definition and Data Types

The Chicago Bike Sharing Project, a significant stride towards sustainable transportation, is designed to enhance community health and accessibility. Its careful consideration of user needs, geographical factors, bicycle types, and operational frameworks ensures that it is not just a project but a service that values and considers its users. This approach has the potential to effectively address urban transportation challenges and ensure the project's sustainability and positive impact on Chicago.

Stage 1: Understanding the User Base

1. User Demographics: The Chicago Bike Sharing Project primarily serves students and commuters in urban areas. By focusing on membership subscriptions, the project sustains itself through minimal profit margins, often relying on external support from local government and stakeholders who benefit from increased urban mobility.
2. Membership Structure: The membership system offers various subscription periods—annual, semi-annual, quarterly, monthly, and weekly—to accommodate user needs. Members benefit from easy access to bicycles without the long-term commitment of purchasing a bike.
3. Geographical Boundaries: The project's operational area is defined by strategic planning, which involves:
    - Assessing the density and distribution of stations based on user demand and traffic density.
    - Prioritizing areas with high foot traffic, such as near universities, workplaces, and residential neighborhoods.
    - Ensuring stations are placed where safety is a concern, particularly avoiding high-crime neighborhoods.
4. Station Distribution: Station locations are determined through data analysis and community feedback. The project assesses popular routes, commuter patterns, and potential barriers to access. Considerations for student routes to schools and universities play a critical role in station placement. Unlike on-campus bike services, the bike-sharing system allows all members to leave bicycles at any designated station.

Stage 2: Bicycle Types and Sharing Mechanism

1. Bicycle Selection: The project needs to determine the types of bicycles available for users. Considerations might include lightweight designs for ease of use, electric bicycles for longer distances, and cargo bikes for carrying items.
2. Ownership and Contribution Programs: Exploring a time-sharing model, where individuals can purchase bicycles with loans and contribute them to the bike-sharing program, could provide benefits. In return, subscribers would receive membership discounts based on the number of bikes offered. This system expands the range of available bicycles and fosters community involvement in the bike-sharing initiative.

Stage 3: System Manual and Documentation

1. Administrative Models

- Contributing Bikes: Develop a standardized process for bike donations, including criteria for bike acceptance, registration, and integration into the fleet.
- Maintenance Schedules: Implement a preventive maintenance schedule to keep bikes in optimal condition. This could involve regular inspections, servicing intervals, and a tracking system for maintenance history.
- User Engagement: Establish strategies to engage users, such as loyalty programs, user feedback surveys, and community events to promote the bike-sharing system.

2. File Management

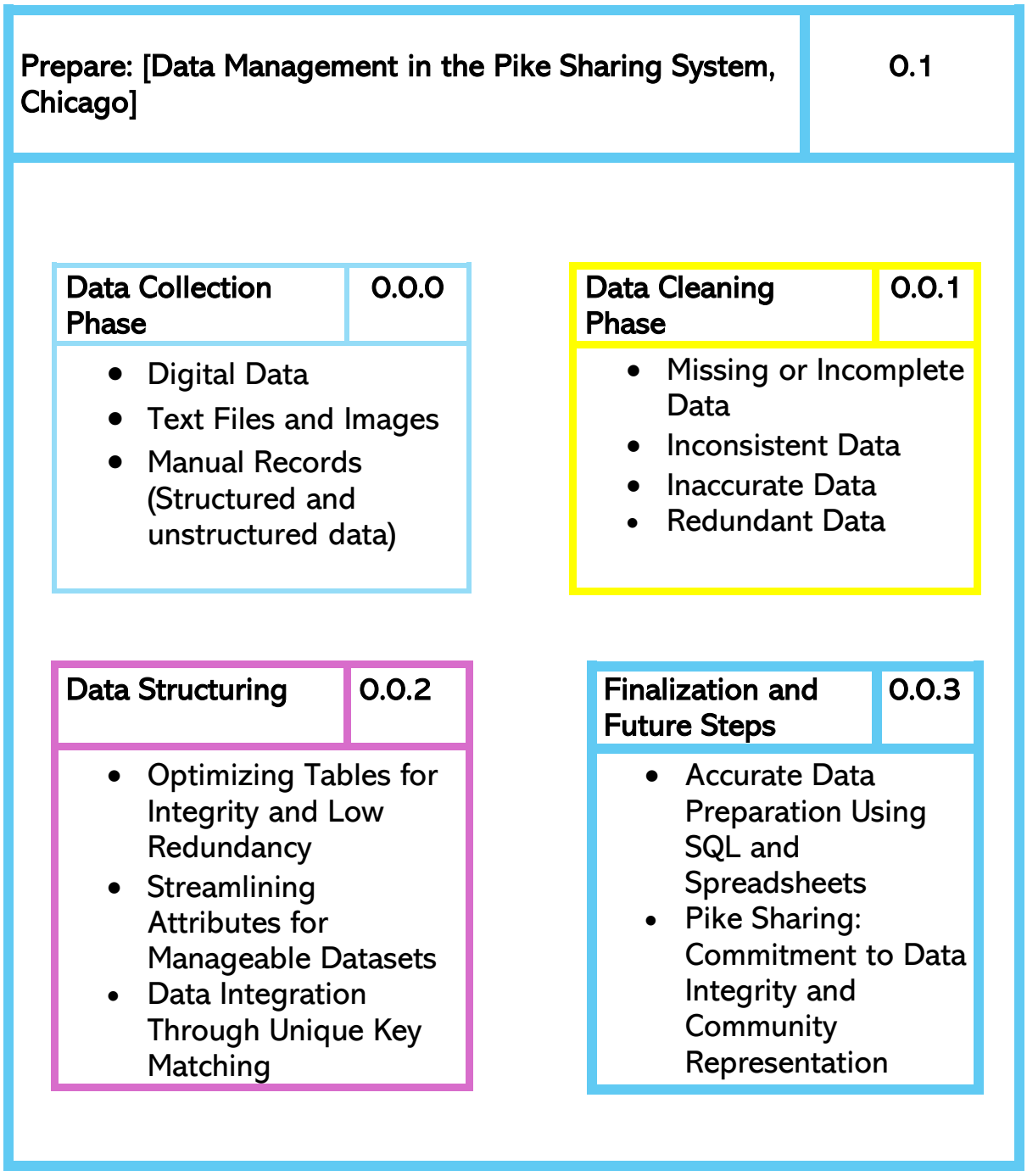
- User Data: Create a secure and efficient system to store and manage user data, ensuring compliance with privacy regulations.
- Maintenance Records: Maintain detailed logs of maintenance activities, including dates, performed tasks, and responsible personnel.
- Membership Statistics: Track membership data, such as new registrations, active users, and usage patterns, to analyze and improve the service.
- Feedback: Collect and document user feedback regularly to identify strengths and areas for improvement.

3. Additional Considerations

- Safety Protocols: Outline safety protocols for users, including proper bike usage and emergency contact information.

- Training Programs: Develop training programs for staff to ensure they are well-versed in maintenance procedures and customer service.
- Sustainability Initiatives: Implement sustainability measures, such as recycling old parts, using eco-friendly materials, and promoting environmental awareness.

0.1. Prepare Phase [4 Stages]



This phase contains four stages.

Phase 2: Data Management in the Pike Sharing System, Chicago

The Pike Sharing system in Chicago is an initiative to optimize mobility and transport resources within the city. Effective data management is crucial for analyzing patterns, improving services, and ensuring stakeholder satisfaction. The project involves structured data, such as user statistics and vehicle locations, and unstructured data, including text feedback and image captures. By prioritizing data collection, cleaning, and structuring, the project team lays a robust groundwork for analysis and strategic decision-making, ultimately driving the success of mobility solutions across the city.

Stage 1: System Definition and Data Types

1. Digital Data: Accessed from databases containing structured records

- Identify Data Sources: Determine which databases contain relevant information, such as user registration systems, trip logs, and vehicle tracking systems.
- Data Extraction: Use data extraction tools or scripts to pull the necessary records from these databases.
- Data Validation: Verify the accuracy and completeness of the extracted data to ensure reliability.
- Storage: Store the extracted data in a centralized database for easy access and analysis.

2. Text Files and Images: Gathered from community feedback and various reports

- Collect Feedback: Gather text feedback from multiple sources, such as user surveys, social media, and mobile app reviews.
- Compile Reports: Collect various reports, including text files and images documenting user experiences and system performance.
- Data Formatting: Convert the gathered files into a standard format suitable for analysis (e.g., converting handwritten notes to digital text).
- Categorize Data: Organize the data into categories, such as positive feedback, areas for improvement, and visual documentation of issues.
- Store and Index: Store the data in a central repository and create an index for easy retrieval during analysis.

3. Manual Records: Engaged with records maintained outside the central system

- Identify Relevant Records: Locate and identify manual records pertinent to the project, such as historical maintenance and user logs.
- Digitize Records: Convert physical records into a digital format using scanning and optical character recognition (OCR) technology.

- Data Cleaning: Review and clean the digitized data to correct any errors introduced during the conversion process.
- Integrate with Central System: Incorporate the digitized records into the central database, ensuring they are correctly linked with existing digital data.
- Historical Analysis: Use the historical context provided by these records to enhance current data analysis and decision-making.

Stage 2: Data Cleaning and Preparation

1. Missing or Incomplete Data
   - Data Augmentation: Use similar datasets to fill in the gaps.
   - Machine Learning Models: Train models specifically designed to predict missing values based on the rest of your data.
2. Inconsistent Data
   - Standardization: Convert data to a standard format (e.g., date formats, units of measure).
   - Normalization: Uniform text formatting (e.g., all lowercase, removing extra spaces).
   - Regular Expressions: Use regex patterns to find and correct inconsistencies.
3. Inaccurate Data
   - Validation Rules: Implement rules for data entry to catch errors.
   - Cross-Verification: Compare data against trusted sources or benchmarks.
   - Domain Expertise: Leverage expertise to identify and correct inaccuracies.
4. Redundant Data
   - Deduplication: Identify and remove duplicate records.
   - Database Normalization: Structure your database to minimize redundancy.
   - Aggregation: Consolidate data where appropriate.

Stage 3: Data Structuring

1. Optimizing Tables for Integrity and Low Redundancy

- Normalization: Splitting data into related tables minimizes redundancy and improves data integrity.
- Indexing: Creating indexes on frequently searched columns to speed up queries.

- Constraints: To maintain data integrity, apply constraints such as primary and foreign keys, and unique and not-null constraints.

2. Streamlining Attributes for Manageable Datasets

- Attribute Selection: Removing unnecessary attributes to simplify the dataset.
- Data Typing: Ensuring each attribute has the appropriate content type.
- Standardization: Standardizing attributes across datasets to ensure consistency.

3. Data Integration Through Unique Key Matching

- Key Matching: Using unique keys (IDs) to join tables from different sources.
- Data Merging: Merging datasets based on unique keys to integrate data into a comprehensive dataset.
- Handling Duplicates: Identifying and resolving duplicate records that may arise during data integration.

Stage 4: Finalization and Future Steps

1. Accurate Data Preparation Using SQL and Spreadsheets

- SQL: Ensure the data is clean and structured by running SQL queries to validate and organize the data. Use queries to identify and correct inconsistencies, missing values, or inaccuracies.
- Spreadsheets: Utilize spreadsheets for additional data verification and visualization. This involves checking data points, applying filters to identify outliers, and using pivot tables to summarize data.

2. Pike Sharing: Commitment to Data Integrity and Community Representation

- Data Integrity: Commit to maintaining high data integrity throughout the process. Regular audits, validations, and updates ensure the data remains accurate and reliable.
- Community Representation: Share data insights with the community, ensuring transparency and representation. This involves creating accessible reports and visualizations that are easily understood and interpreted.

Data Cleaning in a Bike-Sharing System sample codes in Python and SQL

1. Handling Missing Values
    - Detection: Identify missing values using Pandas in Python or SQL commands.
    - Strategy:
        - Remove rows with missing critical values (e.g., station_id, name, latitude, longitude, and capacity).

- Impute missing values in non-critical columns, such as region_id and last_reported, based on contextual clues or the most frequent value.

Example in Python:

df.dropna(subset=['station_id', 'name', 'latitude', 'longitude', 'capacity'], inplace=True)

df['region_id'].fillna(df['region_id'].mode()[0], inplace=True)

df['last_reported'].fillna(method='ffill', inplace=True)

Example in SQL:

DELETE FROM bike_stations

WHERE station_id IS NULL OR name IS NULL OR latitude IS NULL OR longitude IS NULL OR capacity IS NULL;

UPDATE bike_stations

SET region_id = (SELECT region_id FROM bike_stations GROUP BY region_id ORDER BY COUNT(*) DESC LIMIT 1)

WHERE region_id IS NULL;

2. Correcting Data Types

- Ensure each column has the appropriate data type to facilitate accurate analyses. For example:
- Example in SQL:

ALTER TABLE bike_stations

ALTER COLUMN latitude SET DATA TYPE FLOAT,

ALTER COLUMN longitude SET DATA TYPE FLOAT,

ALTER COLUMN last_reported SET DATA TYPE TIMESTAMP;

3. Removing Duplicates

- Deduplicate the dataset to eliminate redundancy. This can be effectively done using SQL with subqueries to keep the first occurrence of each unique record.

Example in SQL:

DELETE FROM bike_stations

WHERE ctid NOT IN (

```
    SELECT MIN(ctid)
    FROM bike_stations
    GROUP BY station_id, name, latitude, longitude, region_id, rental_method, capacity,
...
);
```

Process	0.2

Ensuring Data Integrity in a Bike-Sharing System	0.2.0
Accuracy Completeness Consistency Trustworthiness	

Handling Data Errors	0.2.1
Data Validation Data Cleaning Data Imputation Data Integration Data Transformation	

Document Cleaning Change	0.2.2
• Documentation Using Changelog • How Related To Data Security	

Phase 1: Ensuring Data Integrity in a Bike-Sharing System

Data integrity in a bike-sharing system is vital for generating accurate insights and making data-driven decisions. By focusing on accuracy, completeness, consistency, and trustworthiness, data becomes a reliable foundation for analysis and operational improvements. This case study explores the methods and practices used to maintain these principles in the data collected from bike-sharing stations in cities like Chicago and New York.

Data Collection

The data comes from structured database files provided by the bike-sharing systems. For instance, the Chicago and New York bike-sharing systems offer tables related to bike stations, including information such as station IDs, names, geographic coordinates (latitude and longitude), rental methods, and capacity.

Stage 1: Accuracy

1. Validation Rules: Implement validation rules to ensure the correctness of data at the time of entry. For example, ensure that latitude and longitude values fall within the expected range for the city.
2. Error Checks: Regularly run error-checking algorithms to detect and correct inaccuracies.

Example in Python:

```
def validate_coordinates(df):
    df['latitude'] = df['latitude'].apply(lambda x: x if -90 <= x <= 90 else None)
    df['longitude'] = df['longitude'].apply(lambda x: x if -180 <= x <= 180 else None)
    return df
df = validate_coordinates(df)
```

Stage 2: Completeness

- Missing Data Handling: Detect and handle missing values appropriately. For critical columns (e.g., station_id, name), rows with missing values should be removed. For non-critical columns (e.g., region_id), imputation can be used.
- Data Audits: Regularly audit data for gaps and take corrective actions.
- Example in Python:

```
df.dropna(subset=['station_id', 'name'], inplace=True)
df['region_id'].fillna(df['region_id'].mode()[0], inplace=True)
```

Stage 3. Consistency

1. Standard Formats: Use standard formats for dates, times, addresses, and other fields. Ensure consistent naming conventions across datasets.
2. Normalization: Normalize data across datasets to ensure that similar data fields have consistent values.

Example in SQL:

```
UPDATE bike_stations
SET region_id = UPPER(region_id);
```

Trustworthiness

1. Data Governance: Establish robust policies to handle data securely and responsibly. Define clear roles and responsibilities for data management.
2. Transparency: Maintain transparency in data processing and management practices. Document data sources, processing steps, and any changes made to the data.

Policy:

Data Governance Policy:

- All data entries must be validated at the source.

- Regular audits will be conducted to ensure data completeness.

- Any discrepancies found during audits must be reported and addressed immediately.

- All changes to the data must be documented, including the reason for the change and the individual responsible.

Phase 2: Handling Data Errors

Stage 1. Data Validation

1. Range Checks: Ensure that numerical values fall within a specified range.

Example of SQL

```
SELECT * FROM bike_stations
WHERE latitude < -90 OR latitude > 90
  OR longitude < -180 OR longitude > 180;
```

2. Format Checks: Verify that data follows a specific format, such as dates and phone numbers.

Example of SQL

```
SELECT * FROM bike_stations
WHERE last_reported !~ '^\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}$';
```

3. Consistency Checks: Ensure related data is consistent across different records.

Example of SQL

```
SELECT * FROM bike_stations
WHERE region_id != UPPER(region_id);
```

4. Uniqueness Checks: Ensure that certain fields contain unique values.

Example of SQL

```
SELECT station_id, COUNT(*)
FROM bike_stations
GROUP BY station_id
HAVING COUNT(*) > 1;
```

5. Dependency Checks: Verify that related fields contain logically consistent values.

Example of SQL

```
SELECT * FROM bike_stations
WHERE capacity <= 0;
```

Stage 2: Data Cleaning

Data cleaning involves identifying and correcting errors in the dataset to improve its quality and ensure reliable analysis. This process includes handling missing values, correcting data types, and removing duplicates. Let's dive into the details: Explained above.

Stage 3: Data Imputation

Data imputation fills in missing values based on other available data.

1. Filling Missing Values with Most Frequent Value:
2. Forward Fill for Time Series Data:

Stage 4: Data Integration

Data integration combines data from multiple sources to create a comprehensive dataset.

1. Merging Datasets:
2. Resolving Conflicts:

Stage 4: Data Transformation

Data transformation converts data into a suitable format for analysis.

1. Standardizing Formats.
2. Normalization.

Phase 3: Document the Cleaning Change

Stage 1: Document Using Changelogs:

is an essential part of the data analysis process. Using changelog provides a transparent, reproducible, accountable, and detailed record of all modifications made to the dataset during the data cleaning process. It ensures that changes can be tracked, understood, and replicated, maintaining data integrity and reliability.

Purpose of Changelog

A changelog is a formal document that records all modifications made to a dataset. It is essential for maintaining the dataset's integrity and ensuring all changes are traceable, transparent, and accountable.

1. Transparency: Open and Detailed Account of All Modifications
    - Open Account: Provides a clear, comprehensive record of changes, including what, how, and why changes were made.
    - Detailed Records: Prevents hidden or undocumented changes, fostering trust and clarity among team members and stakeholders.
2. Reproducibility: Ensures Others Can replicate the Cleaning Process
    - Consistent Methodology: Records exact steps taken, allowing others to replicate the process accurately.
    - Reliable Results: Ensures that different analysts can achieve the same results, validating the data analysis process.
3. Accountability: Tracks Who Made Specific Changes and Why
    - Responsible Changes: Specifies who made each change, promoting responsibility and ownership.
    - Rationale for Changes: Documents reasons behind modifications, ensuring changes are valid and justified.
4. Audit Trail: Maintains a Historical Record for Future Reference or Review
    - Historical Record: Provides a chronological record of modifications, helping track the dataset's history.

- Future Reference: Valuable for reviews, audits, or investigations, offering insights into past changes and their impact.

How Related to Data Security

By rigorously maintaining comprehensive documentation, including detailed changelogs, data security is significantly enhanced through transparency, reproducibility, accountability, and establishing a thorough audit trail. These measures are instrumental in safeguarding the dataset against unauthorized access, ensuring adherence to regulatory requirements, and facilitating forensic analysis during a security breach.

Transparency: Open and Detailed Account of All Modifications

- Access Control: Documentation helps establish clear access controls, preventing unauthorized access and ensuring only authorized personnel can make modifications.
- Incident Tracking: Allows quick identification of changes in case of a security breach, aiding in pinpointing the source and taking corrective actions.

Reproducibility: Ensures Others Can replicate the Cleaning Process

- Consistent Security Practices: Ensures consistent application of security practices like encryption and anonymization across datasets.
- Verification of Changes: Allows verification that no unauthorized changes were made, maintaining the dataset's integrity.

Accountability: Tracks Who Made Specific Changes and Why

- User Accountability: Holds individuals responsible for their actions, discouraging malicious activities and promoting responsible data handling.
- Auditability: Enables thorough audits in case of security incidents, identifying potential lapses and ensuring proper procedures are followed.

Audit Trail: Maintains a Historical Record for Future Reference or Review

- Forensic Analysis: Provides a historical record for forensic analysis, helping investigators understand the sequence of events leading to a breach.
- Compliance and Regulation: Helps organizations comply with regulatory requirements and demonstrate commitment to data security practices.

Analyze (How to organize and format the data)	0.3

Data Organization	0.3.0

1. Data Collection
2. Data Cleaning
3. Data Structuring
4. Data Integration
5. Data Storage

Format and adjust the Data	0.3.1

- Reformatting Data
- Data Validation and Conditional Format
- Data Transformation
- Handling Outliers
- Data Imputation
- Derived Variables
- Data Subsetting

Aggregate Data	0.3.2

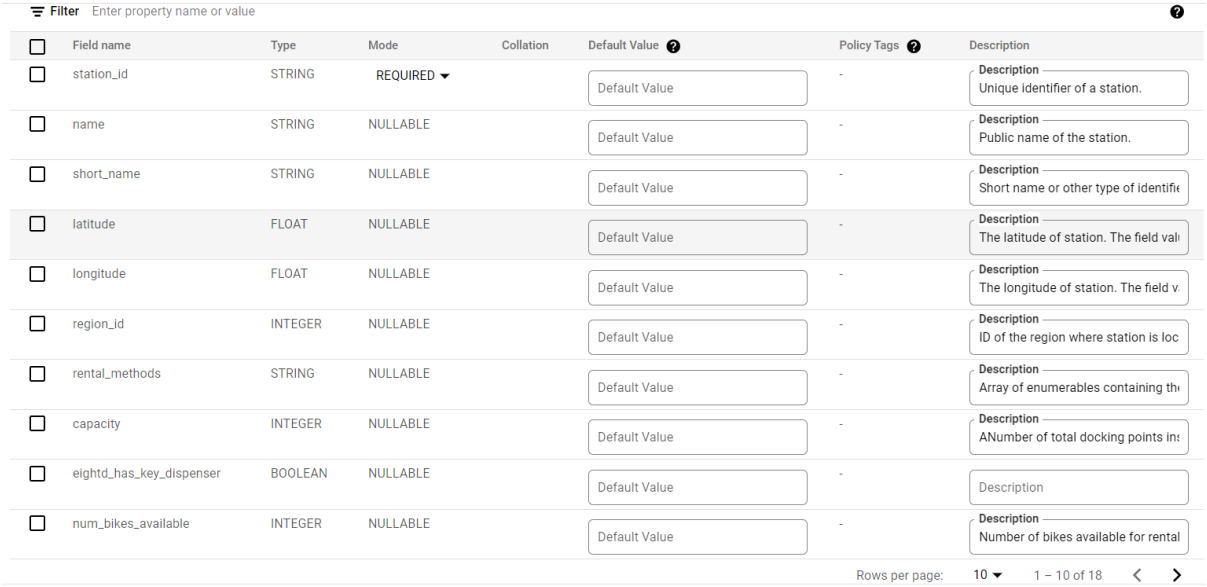
- Identify Trends
- **Make Comparisons**
- **Gain insights**

Data calculations (Transform)	0.3.3

- Formulas and Functions
- Pivot Tables

Some tables of the database as a model

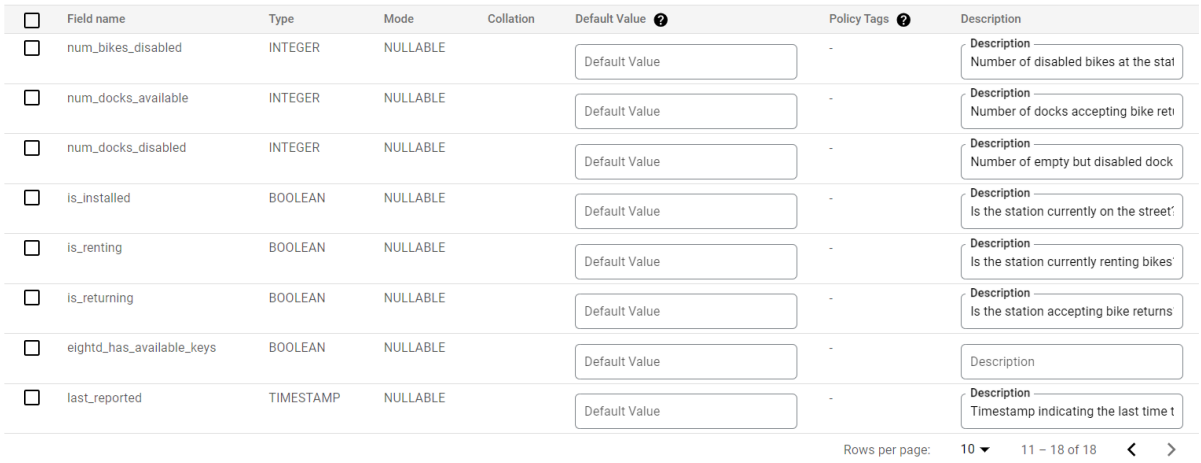
Citebike_Stations



Filter Enter property name or value

Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
station_id	STRING	REQUIRED		Default Value	-	Description: Unique identifier of a station.
name	STRING	NULLABLE		Default Value	-	Description: Public name of the station.
short_name	STRING	NULLABLE		Default Value	-	Description: Short name or other type of identifie
latitude	FLOAT	NULLABLE		Default Value	-	Description: The latitude of station. The field val
longitude	FLOAT	NULLABLE		Default Value	-	Description: The longitude of station. The field v
region_id	INTEGER	NULLABLE		Default Value	-	Description: ID of the region where station is loc
rental_methods	STRING	NULLABLE		Default Value	-	Description: Array of enumerables containing th
capacity	INTEGER	NULLABLE		Default Value	-	Description: ANumber of total docking points ins
eightd_has_key_dispenser	BOOLEAN	NULLABLE		Default Value	-	Description
num_bikes_available	INTEGER	NULLABLE		Default Value	-	Description: Number of bikes available for rental

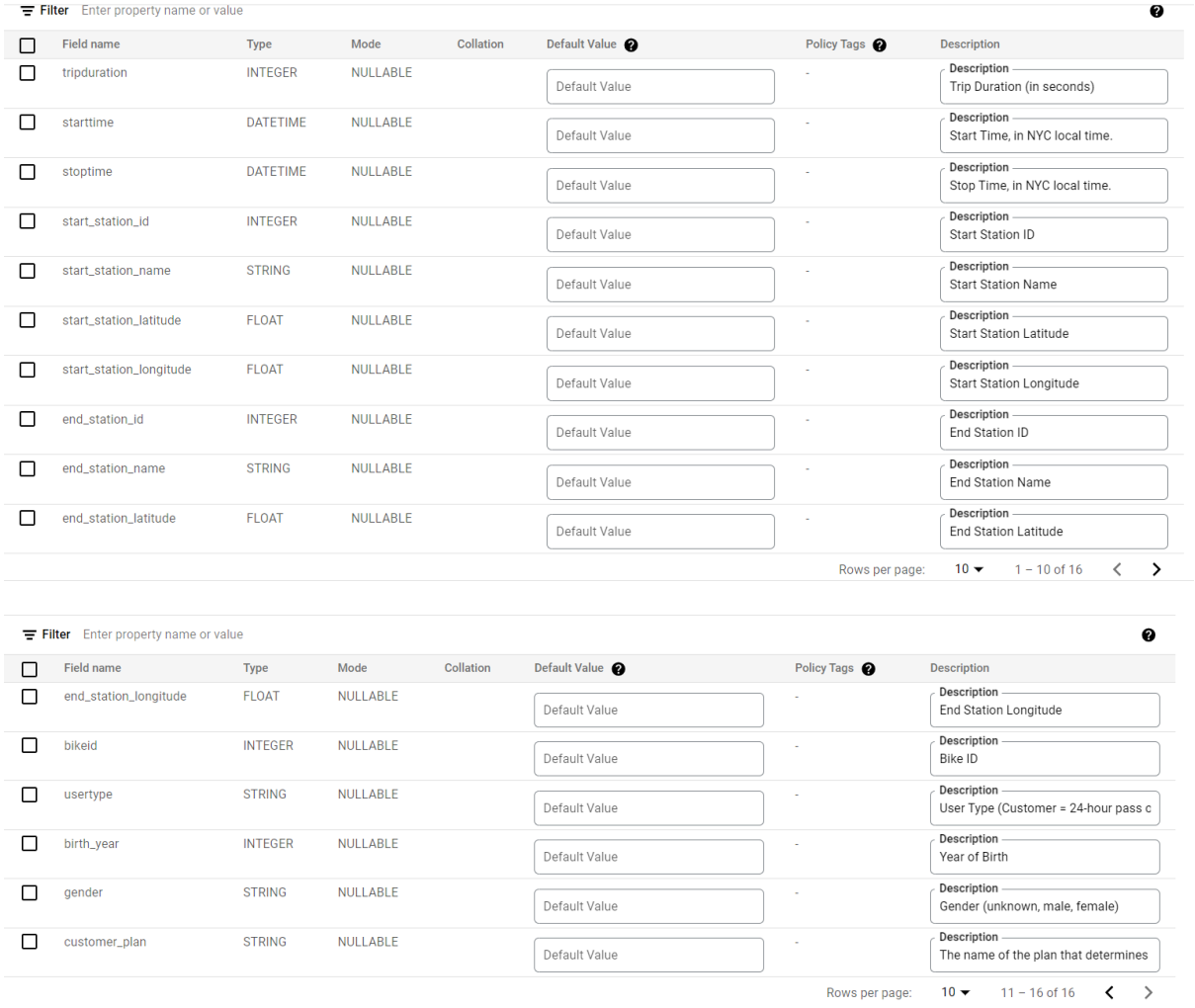
Rows per page: 10 1 – 10 of 18



Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
num_bikes_disabled	INTEGER	NULLABLE		Default Value	-	Description: Number of disabled bikes at the stat
num_docks_available	INTEGER	NULLABLE		Default Value	-	Description: Number of docks accepting bike reti
num_docks_disabled	INTEGER	NULLABLE		Default Value	-	Description: Number of empty but disabled dock
is_installed	BOOLEAN	NULLABLE		Default Value	-	Description: Is the station currently on the street
is_renting	BOOLEAN	NULLABLE		Default Value	-	Description: Is the station currently renting bikes
is_returning	BOOLEAN	NULLABLE		Default Value	-	Description: Is the station accepting bike returns
eightd_has_available_keys	BOOLEAN	NULLABLE		Default Value	-	Description
last_reported	TIMESTAMP	NULLABLE		Default Value	-	Description: Timestamp indicating the last time t

Rows per page: 10 11 – 18 of 18

Citebike_Trips



Filter Enter property name or value

Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
tripduration	INTEGER	NULLABLE		Default Value	-	Trip Duration (in seconds)
starttime	DATETIME	NULLABLE		Default Value	-	Start Time, in NYC local time.
stoptime	DATETIME	NULLABLE		Default Value	-	Stop Time, in NYC local time.
start_station_id	INTEGER	NULLABLE		Default Value	-	Start Station ID
start_station_name	STRING	NULLABLE		Default Value	-	Start Station Name
start_station_latitude	FLOAT	NULLABLE		Default Value	-	Start Station Latitude
start_station_longitude	FLOAT	NULLABLE		Default Value	-	Start Station Longitude
end_station_id	INTEGER	NULLABLE		Default Value	-	End Station ID
end_station_name	STRING	NULLABLE		Default Value	-	End Station Name
end_station_latitude	FLOAT	NULLABLE		Default Value	-	End Station Latitude

Rows per page: 10 1 – 10 of 16

Filter Enter property name or value

Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
end_station_longitude	FLOAT	NULLABLE		Default Value	-	End Station Longitude
bikeid	INTEGER	NULLABLE		Default Value	-	Bike ID
usertype	STRING	NULLABLE		Default Value	-	User Type (Customer = 24-hour pass c
birth_year	INTEGER	NULLABLE		Default Value	-	Year of Birth
gender	STRING	NULLABLE		Default Value	-	Gender (unknown, male, female)
customer_plan	STRING	NULLABLE		Default Value	-	The name of the plan that determines

Rows per page: 10 11 – 16 of 16

other files like Membership_Typew, Personal_info, Bikes_Types, Crashs… etc…

Data is the cornerstone of modern analysis and decision-making. Across various fields, such as business, healthcare, and research, the quality and usability of data directly influence the accuracy and relevance of the insights derived. This guide provides a comprehensive overview of the essential steps to prepare data for analysis, ensuring it is clea n, structured, and ready for insightful exploration and interpretation.

Stage1: Data Collection

Objective: Gather all relevant data from various sources.

- Internal Databases: Collect data from your organization's databases and data warehous es.
- External Data Sources: Import data from third-party providers or public databases.
- Manual Entries: Gather data from surveys, forms, or other manual methods.

Stage 2: Data Cleaning

Objective: Remove or correct errors, inconsistencies, and duplicates to ensure data quality.

- Removing Duplicate records: Identify and eliminate duplicate records.
- Handling Missing Values: Fill in, interpolate, or remove missing data points.
- Correcting Errors: Fix incorrect entries, such as misspelled product names or incorrect dates.

3. Data Structuring

Objective: Arrange data in a structured format for easy analysis.

- Normalization: Organize data into tables to reduce redundancy and ensure data integri ty.
- Categorization: Classify data into broader categories for better management.
- Creating Relationships: Define relationships between data entities for better analysis.

4. Data Integration

Objective: Combine data from different sources into a unified dataset.

- Merging Datasets: Combine internal and external data to create a comprehensive datas et.
- Resolving Conflicts: Harmonize data formats and resolve conflicts.
- Ensuring Consistency: Standardize data types, units of measurement, and formats.

5. Data Storage

Objective: Efficiently store the organized data for easy access and analysis.

- Database Management Systems (DBMS): Use relational databases like MySQL or Po stgreSQL.
- Cloud Storage: Utilize cloud storage solutions such as AWS, Google Cloud, or Azure
- Backup Plans: Implement robust backup plans to ensure data safety and prevent loss.

Phase 2: Format and Adjust the Data

Stage 1: Reformatting Data

- Adjusting Data Types: Ensure the data for each field aligns with its intended format. For instance, converting latitude and longitude fields to FLOAT if not already in this format ensures the station_id remains STRING as it's a unique identifier.
- Standardizing Text Fields: Make necessary text data adjustments, such as converting station names to a consistent format (e.g., all uppercase or lowercase).

Example of SQL

```
ALTER TABLE station_data
MODIFY COLUMN latitude FLOAT,
MODIFY COLUMN longitude FLOAT,
MODIFY COLUMN station_id STRING;
UPDATE station_data
SET name = UPPER(name);
```

Data Validation and Conditional Format

- Check Constraints: Implement checks to ensure data validity, like verifying that the la titude is between -90.0 and 90.0 and longitude is between -180.0 and 180.0.
- Conditional Formatting in Tools: Use visualization tools like spreadsheets or data visu alization software to highlight abnormal or critical values, e.g., showing stations with zero capacity in red.

Example of SQL

```
ALTER TABLE station_data
ADD CONSTRAINT chk_latitude
CHECK (latitude BETWEEN -90.0 AND 90.0),
ADD CONSTRAINT chk_longitude
CHECK (longitude BETWEEN -180.0 AND 180.0);
```

Data Transformation

- Normalizing Data: Convert capacity data into a standardized format if needed, making it comparable across different stations.
- Derived Attributes: Create new attributes like geographical zones or regions based on latitude and longitude, enhancing the data's usability.

Example of SQL

```
ALTER TABLE station_data
ADD zone STRING;
UPDATE station_data
SET zone = CASE
   WHEN latitude > 40.0 THEN 'North'
   ELSE 'South'
END;
```

Handling Outliers

- Identifying Outliers: Detect outliers with statistical methods, such as stations with abn ormally high or low bike capacities.
- Treating Outliers: Decide on the appropriate treatment, such as removing these station s from the dataset or adjusting their values.

```
DELETE FROM station_data
WHERE capacity > (SELECT AVG(capacity) + 3 * STDDEV(capacity) FROM
station_data);
```

Example of SQL

```
DELETE FROM station_data
WHERE capacity > (SELECT AVG(capacity) + 3 * STDDEV(capacity) FROM
station_data);
```

Data Imputation

- Handling Missing Data: Fill in missing values, such as using the average capacity for stations with null values or imputing default values for missing latitude and longitude

```
UPDATE station_data
SET capacity = (SELECT AVG(capacity) FROM station_data)
WHERE capacity IS NULL;
```

Derived Variables

- Creating New Variables: Develop new variables that provide additional insights. For i nstance, a bikes_available_ratio derived from num_bikes_available divided by capa_c ity

```
ALTER TABLE station_data
ADD bikes_available_ratio FLOAT;
UPDATE station_data
SET bikes_available_ratio = num_bikes_available / capacity;
```

Data Subsetting

- Filtering Data: Extract relevant subsets, like stations with a certain number of bikes av ailable or certain regions for targeted analysis.

```
CREATE TABLE north_stations AS
SELECT * FROM station_data
WHERE zone = 'North';
```

Phase 3: Data Aggregation

Through aggregated data analysis, trend identification, comparative evaluation, and insight ge neration, the bike-sharing system's operational efficiency and user experience are remarkably improved. A data-centric approach facilitates informed decision-making, effectively addresses demand fluctuations, and optimizes resource distribution throu ghout the network.

Identify Trends

We will calculate aggregate metrics over time or other groupings to identify trends. This helps us observe patterns and changes in data.

Make Comparisons

Comparisons are vital for understanding how different groups or categories perform relative t o each other. This can highlight strong and weak points.

Example SQL: Compare Average Bikes Available by Region

```
SELECT
   region_id,
   AVG(num_bikes_available) AS avg_bikes_available,
```

```
    AVG(capacity) AS avg_capacity
FROM
    station_data
GROUP BY
    region_id
ORDER BY
    avg_bikes_available DESC;
```

Gain Insights

Gaining insights involves deep-diving into the data to uncover hidden patterns, correlations, or anomalies that could drive decisions and actions.

```
SELECT
    start_station_name,
    COUNT(start_station_id) AS total_rides
FROM
    divvytrip_staging
GROUP BY
    start_station_name
ORDER BY
    total_rides DESC;
```

1. Formulas and Functions

Formulas and functions are essential tools for performing calculations and data transformations. Here are a few commonly used formulas and functions:

Mathematical Functions:

SUM

SELECT SUM(column_name) FROM table_name;

AVG, MAX/MIN, ROUND.

2. String Functions:

CONCAT

SELECT CONCAT(column1, column2) AS combined_column FROM table_name;

UPPER/LOWER, and SUBSTRING

3. Date Functions:

DATEDIFF

SELECT DATEDIFF(end_date, start_date) FROM table_name;

DATE_ADD/DATE_SUB, and EXTRACT

Pivot Tables

Creating a Pivot Table:

Aggregate data based on categories, perform calculations and present the data in a summarize d format.

Example

```
SELECT
   region_id,
   station_id,
   SUM(num_bikes_available) AS total_bikes,
   AVG(capacity) AS avg_capacity
FROM
   station_data
GROUP BY
   region_id, station_id
ORDER BY
   region_id, total_bikes DESC;
```

1. Using Pivot Tables in Excel:
    - Data Selection: Select the data range you want to pivot.
    - Creating Pivot Table:
        a. Go to the "Insert" tab.
        b. Click on "Pivot Table."
        c. Choose the data range and the location for the pivot table.
    - Fields and Areas: Drag and drop fields into the "Rows," "Columns," "Values," and "Filters" areas to customize the pivot table.
2. Examples of Pivot Table Operations:
    - Summarize by Count: Count the number of trips by different stations.

```
SELECT
   start_station_name,
   COUNT(*) AS trip_count
FROM
   divvytrip_staging
GROUP BY
   start_station_name
ORDER BY
   trip_count DESC;
```

- Summarize by Average: Calculate the average trip duration.

```
SELECT
  start_station_name,
  AVG(TIMESTAMPDIFF(MINUTE, started_at, ended_at)) AS avg_duration
FROM
  divvytrip_staging
GROUP BY
  start_station_name
ORDER BY
  avg_duration DESC;
```

- Grouping and Filtering: Group data by region and filter for specific conditions (e.g., o nly showing stations with high bike availability).

Benefits of Using Formulas, Functions, and Pivot Tables

- Efficiency: Quickly summarize large datasets.
- Flexibility: Easily adjust the pivot table to view different aspects of the data.
- Insightful: Gain deeper insights into data trends and patterns.

Share: Transforming Insights into Action	0.3

Data Visualization	0.4.0

- Choosing the Right Chart Types
- Color Schemes and Labels
- Interactivity

Reporting	0.4.1

- Executive Summary
- Detailed Sections
- Appendices

Presentation	0.4.2

- Content
- Engagement
- Visual Aids

Documentation	0.4.3

- Methodology
- Assumptions
- Limitations

Feedback	0.4.4

- Stakeholder Engagement
- Iterative Improvement
- Action Plans

Share: Transforming Insights into Action

The main goal of the sharing phase is to ensure that the insights from data analysis are conveyed to decision-makers, team members, or other stakeholders so they can act on them effectively.

Stage 1: Data Visualization

Choosing the Right Chart Types

Selecting the appropriate chart type is crucial for conveying your data accurately and effectively.

- Bar Charts: Best for comparing discrete categories.
- Line Charts: Ideal for showing trends over time.
- Scatter Plots: Useful for showing the relationship between two variables.
- Pie Charts: Good for showing proportions or percentages.
- Heatmaps: Effective for showing data density or intensity.

Color Schemes and Labels

Using appropriate colors and labels enhances readability and ensures your audience understands your visualizations.

- Consistent Color Schemes: Stick to a consistent color palette that aligns with your data.
- Enhancing Contrast: Ensure there is sufficient contrast between different data points.
- Descriptive Labels: Clearly label axes and data points.

Interactivity

Adding interactivity to your visualizations can make them more engaging and informative.

- Hover Tooltips: Provide additional information on data points when the user hovers over them.
- Clickable Elements: Allow users to drill down into specific data points for more detailed information.
- Interactive Filters: Enable users to filter the data being displayed.

Example Visualization

Tools

Tableau is great for creating interactive and shareable dashboards. It's user-friendly and can handle large datasets. Power BI Integrates well with Microsoft products and is excellent for

creating detailed reports and dashboards. Matplotlib is a Python library for creating static, interactive, and animated visualizations.

Stage 2: Reporting

1. Executive Summary
    - Overview: Briefly describe the purpose and scope of the analysis.
    - Key Findings: Summarize the most important results.
    - Recommendations: Provide concise, actionable suggestions based on the findings.
2. Detailed Sections
    - Introduction: Provide background information and context.
    - Data Collection: Describe the data sources and how the data was collected.
    - Data Cleaning and Preparation: Explain the steps taken to clean and prepare the data for analysis.
    - Exploratory Data Analysis (EDA): Share initial findings from basic data exploration.
    - Modeling and Analysis: Detail the statistical methods and models used.
    - Results: Present the findings from your analysis.
    - Discussion: Interpret the results and provide context.
3. Appendices
    - Data Tables: Include raw data tables or summaries.
    - Technical Documentation: Provide additional details on methods or models used.
    - Additional Visualizations: Share extra charts or graphs that support your findings.

Stage 3: Presentation

1. Content
    - Key Insights: Focus on the most significant findings and insights from your analysis.
    - Structure: Organize the presentation in a logical flow, starting with an introduction, followed by the main body, and ending with a conclusion.
    - Relevance: Tailor the content to the audience's interests and knowledge level.
2. Engagement
    - Storytelling: Use a narrative style to make the data more relatable and engaging.

- Interactive Elements: Incorporate Q&A sessions, polls, or discussions to maintain audience interest.
- Visual Elements: Visual metaphors or analogies help explain complex data points.

3. Visual Aids
    - Slides: Use well-designed slides to support your spoken content, but avoid overloading them with too much text.
    - Infographics: Create concise and visually appealing infographics to illustrate key points.
    - Demonstrations: Where applicable, use live demonstrations or walkthroughs to show how the data analysis works.

Stage 4: Documentation

Methodology

- Steps: Clearly outline the procedures and steps followed during the analysis.
    - Data Collection: Describe the sources and methods used to gather data.
    - Data Cleaning: Detail the steps taken to clean and preprocess the data.
    - Data Analysis: Explain the analytical techniques and tools used.
    - Modeling: Describe the statistical or machine learning models applied.
    - Validation: Outline the processes used to validate the results.

Assumptions

- Data Assumptions: Note any assumptions made about the data, such as completeness or accuracy.
- Model Assumptions: Outline the assumptions underlying any models used, such as linearity or normality.
- Analytical Assumptions: Mention any assumptions regarding the analytical methods, such as independence of observations.

Limitations

- Data Limitations: Discuss any limitations related to the data, such as missing values or potential biases.
- Model Limitations: Highlight any weaknesses or restrictions of the models used, such as overfitting or lack of generalizability.
- Analytical Limitations: Note any constraints or limitations in the analysis process, such as small sample sizes or limited scope.

Stage 5: Feedback

1. Stakeholder Engagement
    - Communication: Maintain open and transparent communication channels with stakeholders throughout the analysis process. This involves regular updates and check-ins to keep everyone informed.
    - Meetings: Schedule meetings or workshops to discuss findings and gather feedback. This helps ensure all stakeholders understand the results and feel their input is valued.
    - Q&A Sessions: Organize Q&A sessions to address stakeholders' questions or concerns about the analysis.
2. Iterative Improvement
    - Feedback Loop: Implement a feedback loop where stakeholders provide input on the analysis, and this feedback is used to refine and improve the work.
    - Continuous Updates: Update the analysis regularly based on new data or insights from stakeholders. This keeps the findings relevant and accurate.
    - Review and Revise: Continuously review and revise the analysis to incorporate stakeholder feedback and improve the overall quality.
3. Action Plans
    - Define Actions: Based on the analysis and stakeholder feedback, define clear and actionable steps that are specific, measurable, and time-bound.
    - Assign Responsibilities: To ensure accountability, assign responsibilities for each action item. Identify who will do what and by when.
    - Monitor Progress: Establish a system for monitoring progress on the action plans. This could involve regular check-ins or progress reports to ensure actions are being implemented effectively.

Act: Transformation & Improvement	0.3

Implementation	0.5.0

- Action Plan
- Resource Allocation
- Execution

Monitoring and Evaluation	0.5.1

- Performance Metrics
- Continuous Monitoring
- Feedback Loop

Adjustments and Improvements	0.5.2

- Analyze Feedback
- Iterative Improvements
- Celebrate Successes:

Documentation and Reporting	0.5.3

- Document Changes
- Regular Reporting:
- Post-Implementation Review

ACT Phase in System Analysis

The ACT phase in system analysis is all about taking actionable steps based on the insights gained from the analysis. This phase is crucial for ensuring the analysis findings translate into meaningful improvements or changes. Let's break it down into detail:

1. Implementation
    - Action Plan: Develop a detailed action plan that outlines the steps necessary to implement the analysis's recommendations. This plan should include specific tasks, timelines, and responsible parties.
    - Resource Allocation: Identify and allocate the necessary resources, such as budget, personnel, and technology, to carry out the action plan.
    - Execution: Carry out the planned actions, ensuring that each step is followed and documented. This may involve deploying new technologies, changing processes, or implementing new policies.
2. Monitoring and Evaluation
    - Performance Metrics: Establish key performance indicators (KPIs) to measure the success of the implementation. These metrics should be directly linked to the goals outlined in the action plan.
    - Continuous Monitoring: Regularly monitor the performance metrics to ensure the implementation is on track. Use dashboards and reports to keep stakeholders informed of progress.
    - Feedback Loop: Gather feedback from stakeholders throughout the implementation process. This helps identify any issues or areas for improvement.
3. Adjustments and Improvements
    - Analyze Feedback: Assess the feedback received and determine if any adjustments are needed. This can help improve the implementation process and ensure better outcomes.
    - Iterative Improvements: Adjust the action plan as necessary based on the analysis of feedback and performance metrics. This iterative process ensures continuous improvement.
    - Celebrate Successes: Recognize and celebrate milestones and successes achieved during implementation. This helps maintain morale and motivates the team to continue striving for improvement.

4. Documentation and Reporting
    - Document Changes: Keep detailed records of all actions, changes, and outcomes. This documentation is helpful for future reference and audits.
    - Regular Reporting: Create regular reports to share progress with stakeholders. These reports should highlight key achievements, challenges, and next steps.
    - Post-Implementation Review: Conduct a thorough review after implementation to evaluate the overall success and identify lessons learned. Use this review to inform future projects and actions.