Bikesharing - Divvybikes
Documentation of the data analysis process

1. Import raw data into local MySql database

1. Import of the last 12 months
2. One separate table for each month
3. Splitting raw csv files into smaller chunks due to upload timeouts

```
split -l 150000  *-divvy-tripdata.csv 202403-chunk_;
rename 's/$/.csv/' *chunk*
```

4. Database table scheme

```
CREATE TABLE `month_YYYY_MM` (
  `ride_id` varchar(16) DEFAULT NULL,
  `rideable_type` varchar(13) DEFAULT NULL,
  `started_at` varchar(19) DEFAULT NULL,
  `ended_at` varchar(19) DEFAULT NULL,
  `start_station_name` varchar(250) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  `start_station_id` varchar(35) DEFAULT NULL,
  `end_station_name` varchar(250) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  `end_station_id` varchar(250) DEFAULT NULL,
  `start_lat` varchar(18) DEFAULT NULL,
  `start_lng` varchar(18) DEFAULT NULL,
  `end_lat` varchar(18) DEFAULT NULL,
  `end_lng` varchar(18) DEFAULT NULL,
  `member_casual` varchar(6) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- identical table for deleted / inconsistent data
CREATE TABLE `bikedata_deleted` (
  `ride_id` varchar(16) DEFAULT NULL,
  `rideable_type` varchar(13) DEFAULT NULL,
  `started_at` varchar(19) DEFAULT NULL,
  `ended_at` varchar(19) DEFAULT NULL,
  `start_station_name` varchar(250) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  `start_station_id` varchar(35) DEFAULT NULL,
  `end_station_name` varchar(250) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  `end_station_id` varchar(250) DEFAULT NULL,
  `start_lat` varchar(18) DEFAULT NULL,
  `start_lng` varchar(18) DEFAULT NULL,
  `end_lat` varchar(18) DEFAULT NULL,
  `end_lng` varchar(18) DEFAULT NULL,
  `member_casual` varchar(6) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

5. Import Query

```
LOAD DATA LOCAL INFILE '~/202306-chunk_aa.csv'
INTO TABLE month_YYYY_MM
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

2. Data manipulation

1. Merge monthly data from tables into one summary table

```
-- repeat query for every single month table
INSERT INTO
   bikedata
SELECT * FROM month_YYYY_MM;
```

2. Update missing data to NULL values

```
UPDATE
   bikedata
SET
   start_station_name = NULL
WHERE
   start_station_name = "";

UPDATE
   bikedata
SET
   start_station_id = NULL
WHERE
   start_station_id = "";

UPDATE
   bikedata
SET
   end_station_name = NULL
WHERE
   end_station_name = "";

UPDATE
   bikedata
SET
   end_station_id = NULL
WHERE
   end_station_id = "";
```

3. Add new column showing the ride duration based on start / end time

```
ALTER TABLE
    bikedata
ADD
    `ride_length` INT NULL AFTER `ended_at`;

UPDATE
    bikedata
SET
    ride_length = TIMESTAMPDIFF(SECOND, started_at, ended_at);
```

4. Add new column showing the day of week as number (1 = Sunday, 7 = Saturday)

```
ALTER TABLE
    bikedata
ADD
    `day_of_week` INT NULL AFTER `ride_length`;

UPDATE
    bikedata
SET
    day_of_week = DAYOFWEEK(started_at);
```

5. Create indexes

```
ALTER TABLE bikedata
    ADD UNIQUE KEY `ride_id` (`ride_id`),
    ADD KEY `rideable_type` (`rideable_type`),
    ADD KEY `ride_length` (`ride_length`),
    ADD KEY `day_of_week` (`day_of_week`);
```

3. Data integrity checks

1. Inspect rideable_types

```
SELECT
    DISTINCT rideable_type
FROM
    bikedata;
-- results in: classic_bike, docked_bike, electric_bike
```

2. Inspect member_casual

```
SELECT
    DISTINCT member_casual
FROM
    bikedata;
-- results in: member, casual
```

3. Check minimum and maximum date values

```
-- ordering also would show NULL values
SELECT * FROM `bikedata` ORDER BY ended_at ASC;
SELECT * FROM `bikedata` ORDER BY ended_at DESC;
SELECT * FROM `bikedata` ORDER BY started_at ASC;
SELECT * FROM `bikedata` ORDER BY started_at DESC;
-- all datetime values were within expected values
-- no missing datetime values
```

4. Identify and delete wrong datetime values (start not before end)

```
SELECT
    *
FROM
    bikedata
WHERE
    started_at >= endet_at;

-- copy inkonsistent data into separate table and delete from source table
INSERT INTO
    bikedata_deleted
SELECT
    *
FROM
    bikedata
WHERE
    started_at >= endet_at;

-- delete
DELETE FROM
    bikedata
WHERE
    started_at >= endet_at;
```

5. Count lines with missing data

```
SELECT
    COUNT(*)
FROM
    bikedata
WHERE
    start_station_name IS NULL
OR
    start_station_id IS NULL
OR
    end_station_name IS NULL
OR
    end_station_id IS NULL;
-- 1.386.978 lines contain NULL values in datetime columns
```

4. Analysis

1. Calculate mean ride duration values by member group

```
-- overall
SELECT
    ROUND(AVG(ride_length)) AS avg_ride_length
FROM
    bikedata;

-- member, casual
SELECT
    member_casual,
    ROUND(AVG(ride_length)) AS avg_ride_length
FROM
    bikedata
GROUP BY
    member_casual;
```

2. Calculate median ride duration values by member group

```
-- overall
SET @rowindex := -1;
SELECT
    ROUND(AVG(b.ride_length)) as median
FROM
    (
    SELECT @rowindex:=@rowindex + 1 AS rowindex,
          ride_length
    FROM
        bikedata
    ORDER BY
        ride_length
    ) AS b
WHERE
    b.rowindex IN (FLOOR(@rowindex / 2), CEIL(@rowindex / 2));

-- member
SET @rowindex := -1;
SELECT
    ROUND(AVG(b.ride_length)) as median
FROM
    (
    SELECT @rowindex:=@rowindex + 1 AS rowindex,
          ride_length
    FROM
        bikedata
    WHERE
        member_casual = "member"
    ORDER BY
        ride_length
    ) AS b
WHERE
    b.rowindex IN (FLOOR(@rowindex / 2), CEIL(@rowindex / 2));
```

```
-- casual
SET @rowindex := -1;
SELECT
   ROUND(AVG(b.ride_length)) as median
FROM
   (
   SELECT @rowindex:=@rowindex + 1 AS rowindex,
         ride_length
   FROM
      bikedata
   WHERE
      member_casual = "casual"
   ORDER BY
      ride_length
   ) AS b
WHERE
   b.rowindex IN (FLOOR(@rowindex / 2), CEIL(@rowindex / 2));
```

3. Calculate ride duration values by day of week and member group

```
-- member, casual
SELECT
   member_casual,
   day_of_week,
   ROUND(AVG(ride_length)) AS avg_ride_length
FROM
   bikedata
GROUP BY
   member_casual,
   day_of_week
ORDER BY
   member_casual,
   day_of_week ASC;
```

4. Calculate mean ride duration values by month and member group

```
-- overall
SELECT
   CONCAT(YEAR(started_at), '-', LPAD(MONTH(started_at), 2, 0)) AS mon,
   ROUND(AVG(ride_length)) AS avg_ride_length
FROM
   bikedata
GROUP BY
   mon
ORDER BY
   mon ASC;
```

```
-- member, casual
SELECT
   member_casual,
   CONCAT(YEAR(started_at), '-', LPAD(MONTH(started_at), 2, 0)) AS mon,
   ROUND(AVG(ride_length)) AS avg_ride_length
FROM
   bikedata
GROUP BY
   member_casual,
   mon
ORDER BY
   member_casual,
   mon ASC;
```

5. Calculate median ride duration values by month and member group

```
-- queries have to be executed separately for every single month
-- overall
SET @rowindex := -1;
SELECT
   ROUND(AVG(b.ride_length)) as median
FROM
   (
   SELECT @rowindex:=@rowindex + 1 AS rowindex,
         ride_length
   FROM
      bikedata
   WHERE
      DATE_FORMAT(started_at, "%Y-%m") = "2023-04"
   ORDER BY
      ride_length
   ) AS b
WHERE
   b.rowindex IN (FLOOR(@rowindex / 2), CEIL(@rowindex / 2));

-- member
SET @rowindex := -1;
SELECT
   ROUND(AVG(b.ride_length)) as median
FROM
   (
   SELECT @rowindex:=@rowindex + 1 AS rowindex,
         ride_length
   FROM
      bikedata
   WHERE
      DATE_FORMAT(started_at, "%Y-%m") = "2023-04"
   AND
      member_casual = "member"
   ORDER BY
      ride_length
   ) AS b
WHERE
   b.rowindex IN (FLOOR(@rowindex / 2), CEIL(@rowindex / 2));
```

```
-- casual
SET @rowindex := -1;
SELECT
   ROUND(AVG(b.ride_length)) as median
FROM
   (
   SELECT @rowindex:=@rowindex + 1 AS rowindex,
         ride_length
   FROM
      bikedata
   WHERE
      DATE_FORMAT(started_at, "%Y-%m") = "2023-04"
   AND
      member_casual = "casual"
   ORDER BY
      ride_length
   ) AS b
WHERE
   b.rowindex IN (FLOOR(@rowindex / 2), CEIL(@rowindex / 2));
```

6. Calculate maximum ride duration values by member group

```
-- member, casual
SELECT
   member_casual,
   MAX(ride_length) AS max_ride_length
FROM
   bikedata
GROUP BY
   member_casual;
```

7. Calculate maximum ride duration values by month and member group

```
-- member, casual
SELECT
   member_casual,
   CONCAT(YEAR(started_at), '-', LPAD(MONTH(started_at), 2, 0)) AS mon,
   MAX(ride_length) AS max_ride_length
FROM
   bikedata
GROUP BY
   member_casual,
   mon
ORDER BY
   member_casual,
   mon ASC;
```

8. Calculate count of rides by member group

```
-- member casual
SELECT
   member_casual,
   COUNT(*) AS count_rides
FROM
   bikedata
GROUP BY
   member_casual
ORDER BY
   member_casual;
```

9. Calculate count of rides by month and member group

```
-- overall
SELECT
   DATE_FORMAT(started_at, "%Y-%m") AS mon,
   COUNT(*) AS count_rides
FROM
   bikedata
GROUP BY
   mon
ORDER BY
   mon ASC;

-- member, casual
SELECT
   DATE_FORMAT(started_at, "%Y-%m") AS mon,
   member_casual,
   COUNT(*) AS count_rides
FROM
   bikedata
GROUP BY
   mon,
   member_casual
ORDER BY
   member_casual,
   mon ASC;
```

10. Calculate count of rides by day of week and member group

```
-- overall
SELECT
   day_of_week,
   COUNT(*) AS count_rides
FROM
   bikedata
GROUP BY
   day_of_week
ORDER BY
   day_of_week ASC;
```

```
-- member, casual
SELECT
    member_casual,
    day_of_week,
    COUNT(*) AS count_rides
FROM
    bikedata
GROUP BY
    member_casual,
    day_of_week
ORDER BY
    member_casual,
    day_of_week ASC;
```