Please Indicate below the group number and then past the following material:

(1) the SQL code you have used to create the schema of your database (only create table and alter table statements (if any), not statements for inserting values)

(2) the SQL code of the queries (possibly with an explanation)

(3) the SQL code used for query optimization for HW2. For each query, indicate the un-optimized version and the optimized one. In case the optimization has been realized through indexes, insert the SQL code for the index creation; in case you have modified the schema (e.g., defined constraints, changed the domain of a field, created a view,  constructed a new materialized table, etc.), insert the code you have used for this modification.

-----

**Group number: 30**

**Students: Bellini Giacomo 1970896, Pierini Elisa 1859043**

— DB Structure

# create the schema of the databases

CREATE SCHEMA cyclist;

CREATE SCHEMA cyclistic\_faster;

# create the six tables for each month until june

CREATE TABLE cyclist.january (

     ride\_id VARCHAR(200) primary key NOT NULL,

     rideable\_type VARCHAR(200),

     started\_at datetime,

     ended\_at datetime,

     start\_station\_name varchar(200),

     start\_station\_id varchar(50),

     end\_station\_name varchar(200),

     end\_station\_id varchar(50),

     start\_lat float,

     start\_lng float,

    end\_lat float,

     end\_lng float,

     member\_casual varchar(50)

); # Do it for all six months until june!

# create the six tables for each month until june

CREATE TABLE cyclistic\_faster.january (

     ride\_id VARCHAR(200) primary key NOT NULL,

     rideable\_type VARCHAR(200),

     started\_at datetime,

     ended\_at datetime,

     start\_station\_name varchar(200),

     start\_station\_id varchar(50),

     end\_station\_name varchar(200),

     end\_station\_id varchar(50),

     start\_lat float,

     start\_lng float,

    end\_lat float,

     end\_lng float,

     member\_casual varchar(50)

); # Do it for all six months until june!

# load the csv data inside the corresponding tables (202101 for january, 202102 for february ecc…)

LOAD DATA LOCAL INFILE 'c:/path/to/data/202101-divvy-tripdata.csv'

INTO TABLE  cyclist.january

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

# load the csv data inside the corresponding tables (202101 for january, 202102 for february ecc…)

LOAD DATA LOCAL INFILE 'c:/path/to/data/202101-divvy-tripdata.csv'

INTO TABLE  cyclistic\_faster.january

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

# replace empty string with NULL values for all the tables

UPDATE cyclist.january

SET rideable\_type = NULL

WHERE rideable\_type = '';

# replace empty string with NULL values for all the tables

UPDATE cyclistic\_faster.january

SET rideable\_type = NULL

WHERE rideable\_type = '';

# delete 30% of rows from april table because it has too much records

DELETE FROM cyclist.april WHERE RAND() <= 0.7

DELETE FROM cyclistic\_faster.april WHERE RAND() <= 0.7

— Queries

**-- 1) SELECT ALL CLASSIC BIKE RIDES THAT HAVE A START AND END STATION ON JANUARY**

**– Select all the rides inside cyclist.january table that were done using a classic bike and have start and end station names populated.**

SELECT ride\_id, start\_station\_name, end\_station\_name from cyclist.january where rideable\_type like '%classic\_bike%' and start\_station\_name is not NULL and end\_station\_name is not NULL;

**-- 2) COUNT NUMBER OF RIDES GROUP BY "rideable\_type" IN JUNE AND JANUARY, ORDERED BY MONTH**

**– Do an union all (drop duplicates) between january and june tables counting the number of rides taken with each type of bike and order the results by month**

SELECT rideable\_type, count(rideable\_type) as number\_of\_rides, 'june' as month from cyclist.june GROUP BY rideable\_type

UNION ALL

SELECT rideable\_type, count(rideable\_type) as number\_of\_rides, 'january' as month from cyclist.january

GROUP BY rideable\_type ORDER BY month DESC;

**-- 3) LET'S SEE THE TOTAL NUMBERS OF "casual" AND "member" IN THE START STATION IN JANUARY:**

**– Do a join between january and february table on start\_station\_name column and count the number of memberships for each month with a switch case to distinguish from member and casual type of membership. Select those rows that have start\_station\_name not null and group and order by the same column**

SELECT

  start\_station\_name,

  COUNT(CASE WHEN cyclist.january.member\_casual LIKE '%member%' THEN 1 END) AS number\_of\_memberships\_jan,

  COUNT(CASE WHEN cyclist.january.member\_casual LIKE '%casual%' THEN 1 END) AS number\_of\_casuals\_jan,

  COUNT(CASE WHEN cyclist.february.member\_casual LIKE '%member%' THEN 1 END) AS number\_of\_memberships\_feb,

  COUNT(CASE WHEN cyclist.february.member\_casual LIKE '%casual%' THEN 1 END) AS number\_of\_casuals\_feb

FROM

  cyclist.january

  JOIN cyclist.february USING(start\_station\_name)

WHERE

  start\_station\_name IS NOT NULL

GROUP BY

  start\_station\_name

ORDER BY

  start\_station\_name;

**-- 4) FIND THE MOST FAMOUS RIDES ACROSS ALL 6 MONTHS**

**– Create a view when you performs an union all with all the tables when you count the number rides for each start station name. Then selects from the view summing up all the numbers grouping by station where is not null and take the highest 10**

with most\_famous\_rides as (

SELECT start\_station\_name, end\_station\_name, COUNT(\*) AS ride\_count

FROM january

GROUP BY start\_station\_name, end\_station\_name

UNION ALL

SELECT start\_station\_name, end\_station\_name, COUNT(\*) AS ride\_count

FROM february

GROUP BY start\_station\_name, end\_station\_name

UNION ALL

SELECT start\_station\_name, end\_station\_name, COUNT(\*) AS ride\_count

FROM march

GROUP BY start\_station\_name, end\_station\_name

UNION ALL

SELECT start\_station\_name, end\_station\_name, COUNT(\*) AS ride\_count

FROM april

GROUP BY start\_station\_name, end\_station\_name

UNION ALL

SELECT start\_station\_name, end\_station\_name, COUNT(\*) AS ride\_count

FROM may

GROUP BY start\_station\_name, end\_station\_name

UNION ALL

SELECT start\_station\_name, end\_station\_name, COUNT(\*) AS ride\_count

FROM june

GROUP BY start\_station\_name, end\_station\_name

) SELECT start\_station\_name, end\_station\_name, SUM(ride\_count) AS ride\_count

 FROM most\_famous\_rides WHERE start\_station\_name IS NOT NULL AND

end\_station\_name IS NOT NULL

GROUP BY start\_station\_name, end\_station\_name

ORDER BY ride\_count DESC LIMIT 10;

**-- 5) Find the busiest time of day for each start station**

**– Count the number of rides taken for each start station and datetime in the format specified. This format makes the query computational heavier, then order by the ride\_count in descending order**

SELECT start\_station\_name, DATE\_FORMAT(started\_at, '%H:%m:%s') AS hour, COUNT(\*) AS ride\_count

FROM june WHERE start\_station\_name IS NOT NULL GROUP BY start\_station\_name, hour ORDER BY ride\_count DESC;

**-- 6) Find the longest distance across all months**

**– Performs an union (keep duplicates) with all six tables and select the maximum distance taken by the rides. Keep only those rows that have max\_distance > 0. Order by the maximum distance in descending order**

SELECT ride\_id, round(MAX(SQRT(POW(start\_lat - end\_lat, 2) + POW(start\_lng - end\_lng, 2))),2) AS max\_distance

FROM january

GROUP BY ride\_id

HAVING max\_distance >0

UNION

SELECT ride\_id, round(MAX(SQRT(POW(start\_lat - end\_lat, 2) + POW(start\_lng - end\_lng, 2))),2) AS max\_distance

FROM february

GROUP BY ride\_id

HAVING max\_distance >0

UNION

SELECT ride\_id, round(MAX(SQRT(POW(start\_lat - end\_lat, 2) + POW(start\_lng - end\_lng, 2))),2) AS max\_distance

FROM march

GROUP BY ride\_id

HAVING max\_distance >0

UNION

SELECT ride\_id, round(MAX(SQRT(POW(start\_lat - end\_lat, 2) + POW(start\_lng - end\_lng, 2))),2) AS max\_distance

FROM april

GROUP BY ride\_id

HAVING max\_distance >0

UNION

SELECT ride\_id, round(MAX(SQRT(POW(start\_lat - end\_lat, 2) + POW(start\_lng - end\_lng, 2))),2) AS max\_distance

FROM may

GROUP BY ride\_id

HAVING max\_distance >0

UNION

SELECT ride\_id, round(MAX(SQRT(POW(start\_lat - end\_lat, 2) + POW(start\_lng - end\_lng, 2))),2) AS max\_distance

FROM june

GROUP BY ride\_id

HAVING max\_distance >0

ORDER BY max\_distance desc;

**-- 7) Find the number of rides per station in february and april**

**– Select the number of rides taken in april and february by doing a join on start\_station\_id column, grouping by start\_station\_name and ordering by the number of rides and take the first 10**

SELECT m2.start\_station\_name, COUNT(\*) as num\_rides

FROM february m2

JOIN april m4 ON m2.start\_station\_id = m4.start\_station\_id

GROUP BY m2.start\_station\_name

ORDER BY num\_rides DESC

LIMIT 10;

**-- 8) Find the number of rides that started and ended on each day in February**

**– Create a view doing an union all between the start\_dates and end\_dates for the rides in february. Then, select the date from it and count the number of started rides and ended rides by doing a left join two times on february table, first time on started\_dates and then on end\_dates. Group all by the dates and order in ascending order**

with ride\_dates as(

SELECT ride\_id, started\_at AS ride\_date

FROM february

UNION ALL

SELECT ride\_id, ended\_at AS ride\_date

FROM february)

SELECT DATE(ride\_date) AS ride\_day,

COUNT(DISTINCT start\_rides.ride\_id) AS "started rides",

COUNT(DISTINCT end\_rides.ride\_id) AS "ended rides"

FROM ride\_dates

LEFT JOIN february AS start\_rides ON ride\_dates.ride\_id = start\_rides.ride\_id AND ride\_dates.ride\_date = start\_rides.started\_at

LEFT JOIN february AS end\_rides ON ride\_dates.ride\_id = end\_rides.ride\_id AND ride\_dates.ride\_date = end\_rides.ended\_at

GROUP BY ride\_day

ORDER BY ride\_day ASC;

**-- 9) Count the number of rides by day of the week and member type on June**

**– Select from june table the number of rides taken each day of the week grouping by the membership type and the day of the week. Eventually, order by number\_of rides to see the busiest days of the week**

SELECT member\_casual, DAYNAME(started\_at) as day\_of\_week, COUNT(\*) as num\_rides

FROM june

GROUP BY member\_casual, DAYNAME(started\_at)

ORDER BY num\_rides DESC;

**-- 10) Find the minimum acceptable distance for each ride in February**

**– Calculate the minimum between the end longitude and the start longitude to get the minimum distance taken by the ride and keep only those rows that have a positive value. Order in descending order**

SELECT ride\_id, round(MIN(end\_lng - start\_lng),2) as min\_distance

FROM february

GROUP BY ride\_id

having min\_distance > 0

ORDER BY min\_distance desc

— Optimized Queries

**--  3) LET'S SEE THE TOTAL NUMBERS OF CASUAL AND MEMBER IN THE START STATION IN JANUARY**

**-- Let’s add and index on start\_station\_name column on january and february tables**

ALTER TABLE cyclistic\_faster.january ADD INDEX idx\_start\_station\_name (start\_station\_name);

ALTER TABLE cyclistic\_faster.february ADD INDEX idx\_start\_station\_name (start\_station\_name);

**-- Let’s add and index on member\_casual  column on january and february tables**

ALTER TABLE cyclistic\_faster.january ADD INDEX idx\_member\_casual (member\_casual);

ALTER TABLE cyclistic\_faster.february ADD INDEX idx\_member\_casual (member\_casual);

SELECT

  start\_station\_name,

  COUNT(CASE WHEN cyclistic\_faster.january.member\_casual LIKE '%member%' THEN 1 END) AS number\_of\_memberships\_jan,

  COUNT(CASE WHEN cyclistic\_faster.january.member\_casual LIKE '%casual%' THEN 1 END) AS number\_of\_casuals\_jan,

  COUNT(CASE WHEN cyclistic\_faster.february.member\_casual LIKE '%member%' THEN 1 END) AS number\_of\_memberships\_feb,

  COUNT(CASE WHEN cyclistic\_faster.february.member\_casual LIKE '%casual%' THEN 1 END) AS number\_of\_casuals\_feb

FROM

  cyclistic\_faster.january

  JOIN cyclistic\_faster.february USING(start\_station\_name)

WHERE

  start\_station\_name IS NOT NULL

GROUP BY

  start\_station\_name

ORDER BY

  start\_station\_name;

**-- 4) FIND THE MOST FAMOUS RIDES ACROSS ALL 6 MONTHS**

**– Create a summary table instead of the view, adding index to summary table and select from it**

CREATE TABLE cyclistic\_faster.summary AS

SELECT start\_station\_name, end\_station\_name, COUNT(\*) AS ride\_count, 'january' AS month

FROM cyclistic\_faster.january

GROUP BY start\_station\_name, end\_station\_name

UNION ALL

SELECT start\_station\_name, end\_station\_name, COUNT(\*) AS ride\_count, 'february' AS month

FROM cyclistic\_faster.february

GROUP BY start\_station\_name, end\_station\_name

UNION ALL

SELECT start\_station\_name, end\_station\_name, COUNT(\*) AS ride\_count, 'march' AS month

FROM cyclistic\_faster.march

GROUP BY start\_station\_name, end\_station\_name

UNION ALL

SELECT start\_station\_name, end\_station\_name, COUNT(\*) AS ride\_count, 'april' AS month

FROM cyclistic\_faster.april

GROUP BY start\_station\_name, end\_station\_name

UNION ALL

SELECT start\_station\_name, end\_station\_name, COUNT(\*) AS ride\_count, 'may' AS month

FROM cyclistic\_faster.may

GROUP BY start\_station\_name, end\_station\_name

UNION ALL

SELECT start\_station\_name, end\_station\_name, COUNT(\*) AS ride\_count, 'june' AS month

FROM cyclistic\_faster.june

GROUP BY start\_station\_name, end\_station\_name;

CREATE INDEX summary\_index ON cyclistic\_faster.summary (start\_station\_name, end\_station\_name, ride\_count);

SELECT start\_station\_name, end\_station\_name, SUM(ride\_count) AS ride\_count

FROM cyclistic\_faster.summary

WHERE start\_station\_name is not null and end\_station\_name is not null

GROUP BY start\_station\_name, end\_station\_name

ORDER BY ride\_count DESC

LIMIT 10;

**-- 5) Find the busiest time of day for each start station**

**– Add where clause to filter down the results. Although this result is different from the unoptimized one it explains how big is the impact of the number of records on the computational time of the query**

SELECT start\_station\_name, DATE\_FORMAT(started\_at, '%H:%m:%s') AS hour, COUNT(\*) AS ride\_count

FROM cyclistic\_faster.june where start\_station\_name is not null AND

HOUR(started\_at) >= 8 AND HOUR(started\_at) < 18

GROUP BY start\_station\_name, hour order by ride\_count desc;

**-- 9) Count the number of rides by day of the week and member type on June**

**-- Add index on member\_casual and started\_at column on june to speed up the query**

CREATE INDEX member\_started ON june (member\_casual, started\_at);

SELECT member\_casual, DAYNAME(started\_at) as day\_of\_week, COUNT(\*) as num\_rides

FROM cyclistic\_faster.june

GROUP BY member\_casual, DAYNAME(started\_at)

ORDER BY num\_rides DESC;