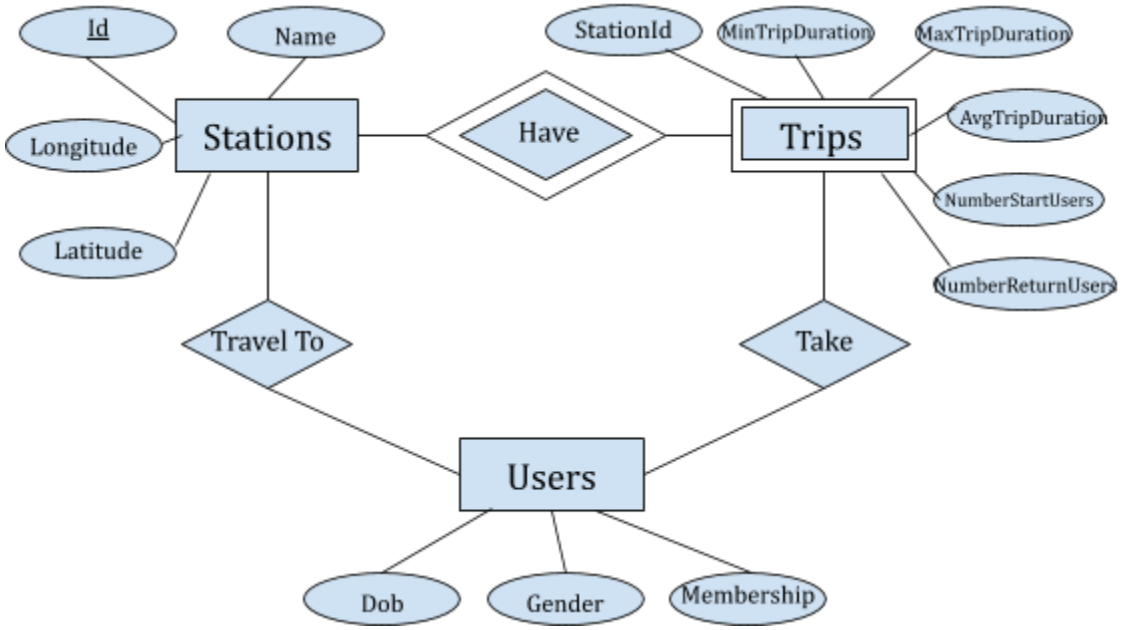
CSc 33600 Introduction to Database Systems
Linda Wong

Assignment 3: CitiBike

[I]

1. Study the data provided for the use of Citi bikes to gain an understanding of the program utilization.
2. Explore the information on the program available on the Web to verify the basic facts and collect additional information regarding the utilization of Citi Bikes. Write down the additional information and/or requirements you gathered. Give the URLs for the websites contacted.
   URL: https://www.citibikenyc.com/system-data
   Confirm Information:
   Gender (Zero=unknown; 1=male; 2=female)
   Duration Time(seconds)
   Usertype(Customer, Subscriber)
3. Draw an appropriate E/R diagram that satisfies the basic and additional facts, indicating weak and subclass entity sets, whenever exist, multiplicity of relationships, and the key, or keys, for each entity set. Distinguish between the parts of the E/R diagram pertaining to the given and additional facts.



DEPENDENCIES:
(start_time, stop_time) → trip_duration
(start_station_id, end_station_id) → StationId
start_station_id → start_station_name
end_station_id → end_station_name
Trip = weak entity

4. Translate the E/R diagram in [3] to a relational database schema.
   Stations(Id, Name, Latitude, Longitude)
   Trips(StationId, MinTripDuration, MaxTripDuration, AvgTripDuration, NumberStartUsers, NumberReturnUsers)
   Users(membership, dob, gender)
5. Specify a number of essential functional dependencies for each relation. Identify possible keys, whenever exist, and the primary key and foreign keys for each relation.
   Primary Key in Stations: Id
   Foreign Key in Trips: StationId

[II]

Create SQL expressions to break the data provided in the file into the following tables:

Stations(Id, Name, Latitude, Longitude)

```
DROP TABLE IF EXISTS CitiBike.Stations;
CREATE TABLE Stations(
        Id INT PRIMARY KEY,
        Name VARCHAR(255),
        Latitude DECIMAL(10,8),
        Longitude DECIMAL(10,8)
);
INSERT INTO CitiBike.Stations
SELECT start_station_id, start_station_name, start_station_latitude, start_station_longitude
FROM CitiBikeData GROUP BY start_station_id
UNION
SELECT end_station_id, end_station_name, end_station_latitude, end_station_longitude
FROM CitiBikeData GROUP BY end_station_id
```

Trips(StationId, MinTripDuration, MaxTripDuration, AvgTripDuration, NumberStartUsers, NumberReturnUsers)

```
DROP TABLE IF EXISTS CitiBike.Trips;
CREATE TABLE Trips(
        StationId INT,
        MinTripDuration INT,
        MaxTripDuration INT,
        AvgTripDuration INT,
        NumberStartUsers INT,
        NumberReturnUsers INT,
        FOREIGN KEY(StationId) REFERENCES Stations(Id)
);
INSERT INTO CitiBike.Trips
SELECT S.StationId, S.MinTripDuration, S.MaxTripDuration, S.AvgTripDuration,
S.NumberStartUsers, E.NumberReturnUsers
```

```
FROM
(SELECT DISTINCT start_station_id AS StationId,
        MIN(trip_duration) AS MinTripDuration,
        MAX(trip_duration) AS MaxTripDuration ,
        FLOOR(AVG(trip_duration)) AS AvgTripDuration,
        COUNT(bike_id) AS NumberStartUsers
        FROM CitiBikeData
        GROUP BY start_station_id
        ORDER BY start_station_id) S
INNER JOIN
(SELECT DISTINCT end_station_id AS StationId,
        COUNT(bike_id) AS NumberReturnUsers
        FROM CitiBikeData
        GROUP BY end_station_id
        ORDER BY end_station_id) E
ON S.StationId = E.StationId
```

UsageByDay(StationId, NumberWeekdayStartUsers, NumberWeekdayReturnUsers,
NumberWeekendStartUsers, NumberWeekendReturnUsers)

```
DROP TABLE IF EXISTS CitiBike.UsageByDay;
CREATE TABLE UsageByDay(
        StationId INT,
        NumberWeekdayStartUsers INT,
        NumberWeekdayReturnUsers INT,
        NumberWeekendStartUsers INT,
        NumberWeekendReturnUsers INT,
        FOREIGN KEY(StationId) REFERENCES Stations(Id)
);
SET SQL_SAFE_UPDATES=0;
UPDATE CitiBikeData
        SET start_day = WEEKDAY(start_time),
                stop_day = WEEKDAY(stop_time);
SET SQL_SAFE_UPDATES=1;
INSERT INTO CitiBike.UsageByDay
SELECT S.StationId, S.NumberWeekdayStartUsers, S.NumberWeekendStartUsers,
          E.NumberWeekdayReturnUsers, E.NumberWeekendReturnUsers
FROM
(SELECT DISTINCT start_station_id AS StationId,
        COUNT(CASE WHEN start_day < 5  THEN bike_id END) AS NumberWeekdayStartUsers,
        COUNT(CASE WHEN start_day >= 5  THEN bike_id END) AS NumberWeekendStartUsers
        FROM CitiBikeData
        GROUP BY start_station_id
        ORDER BY start_station_id) S
```

```
INNER JOIN
(SELECT DISTINCT end_station_id AS StationId,
        COUNT(CASE WHEN stop_day < 5 THEN bike_id END) AS NumberWeekdayReturnUsers,
        COUNT(CASE WHEN stop_day >= 5 THEN bike_id END) AS NumberWeekendReturnUsers
        FROM CitiBikeData
        GROUP BY end_station_id
        ORDER BY end_station_id) E
ON S.StationId = E.StationId
```

UsageByGender(StationId, NumberMaleStartUsers, NumberFemaleStartUsers, NumberMaleReturnUsers, NumberFemaleReturnUsers)

```
DROP TABLE IF EXISTS CitiBike.UsageByGender;
CREATE TABLE UsageByGender(
        StationId INT,
        NumberMaleStartUsers INT,
        NumberFemaleStartUsers INT,
        NumberMaleReturnUsers INT,
        NumberFemaleReturnUsers INT,
        FOREIGN KEY(StationId) REFERENCES Stations(Id)
);
INSERT INTO CitiBike.UsageByGender
SELECT S.StationId, S.NumberMaleStartUsers, S.NumberFemaleStartUsers,
          E.NumberMaleReturnUsers, E.NumberFemaleReturnUsers
FROM
(SELECT DISTINCT start_station_id AS StationId,
        COUNT(CASE WHEN gender = 1 THEN bike_id END) AS NumberMaleStartUsers,
        COUNT(CASE WHEN gender = 2 THEN bike_id END) AS NumberFemaleStartUsers
        FROM CitiBikeData
        GROUP BY start_station_id
        ORDER BY start_station_id) S
INNER JOIN
(SELECT DISTINCT end_station_id AS StationId,
        COUNT(CASE WHEN gender = 1 THEN bike_id END) AS NumberMaleReturnUsers,
        COUNT(CASE WHEN gender = 2 THEN bike_id END) AS NumberFemaleReturnUsers
        FROM CitiBikeData
        GROUP BY end_station_id
        ORDER BY end_station_id) E
ON S.StationId = E.StationId
```

UsageByAge(StationId, NumberMaleUsersUnder18, NumberMaleUsers18To40,
NumberMaleUsersOver40, NumberFemaleUsersUnder18,
NumberFemaleUsers18To40, NumberFemaleUsersOver40)

```
DROP TABLE IF EXISTS CitiBike.UsageByAge;
CREATE TABLE UsageByAge(
        StationId INT,
        NumberMaleUsersUnder18 INT,
        NumberMaleUsers18To40 INT,
        NumberMaleUsersOver40 INT,
        NumberFemaleUsersUnder18 INT,
        NumberFemaleUsers18To40 INT,
        NumberFemaleUsersOver40 INT,
        FOREIGN KEY(StationId) REFERENCES Stations(Id)
);
SET @RECORDYEAR = 2013;
INSERT INTO CitiBike.UsageByAge
SELECT DISTINCT start_station_id AS StationId,
COUNT(CASE WHEN gender = 1 AND (@RECORDYEAR - birth_year < 18) THEN bike_id
END) AS NumberMaleStartUsers,
COUNT(CASE WHEN gender = 1 AND (@RECORDYEAR - birth_year >= 18 AND
@RECORDYEAR - birth_year <= 40) THEN bike_id END) AS NumberMaleUsers18To40,
COUNT(CASE WHEN gender = 1 AND (@RECORDYEAR - birth_year > 40) THEN bike_id
END) AS NumberMaleUsersOver40,
COUNT(CASE WHEN gender = 2 AND (@RECORDYEAR - birth_year < 18) THEN bike_id
END) AS NumberFemaleStartUsers,
COUNT(CASE WHEN gender = 2 AND (@RECORDYEAR - birth_year >= 18 AND
@RECORDYEAR - birth_year <= 40) THEN bike_id END) AS NumberFemaleUsers18To40,
COUNT(CASE WHEN gender = 2 AND (@RECORDYEAR - birth_year > 40) THEN bike_id
END) AS NumberFemaleUsersOver40
FROM CitiBikeData
GROUP BY start_station_id
ORDER BY start_station_id
```

[III]

1. Create appropriate SQL expressions to determine the most frequent trips between any two stations by the day of the week.

```
SET SQL_SAFE_UPDATES=0;
UPDATE CitiBikeData
        SET start_day = WEEKDAY(start_time),
                stop_day = WEEKDAY(stop_time);
SET SQL_SAFE_UPDATES=1;
SELECT DISTINCT
(SELECT start_station_id) AS "Station 1",
(SELECT end_station_id) AS "Station 2",
COUNT(CASE WHEN start_day = 0  THEN bike_id END) AS Number_of_Trips_on_Monday,
COUNT(CASE WHEN start_day = 1  THEN bike_id END) AS "Number of Trips on Tuesday",
COUNT(CASE WHEN start_day = 2  THEN bike_id END) AS "Number of Trips on
Wednesday",
COUNT(CASE WHEN start_day = 3  THEN bike_id END) AS "Number of Trips on Thursday",
COUNT(CASE WHEN start_day = 4  THEN bike_id END) AS "Number of Trips on Friday",
COUNT(CASE WHEN start_day = 5  THEN bike_id END) AS "Number of Trips on Saturday",
COUNT(CASE WHEN start_day = 6  THEN bike_id END) AS "Number of Trips on Sunday"
FROM CitiBikeData
GROUP BY start_station_id, end_station_id
ORDER BY Number_of_Trips_on_Monday DESC
```

Output

Station 1	Station 2	Number_of_Trips_on_Monday	Number of Trips on Tuesday	Number of Trips on Wednesday	Number of Trips on Thursday	Number of Trips on Friday	Number of Trips on Saturday	Number of Trips on Sunday
2006	2006	231	255	217	141	255	467	533
281	281	108	85	95	99	80	249	193
387	387	66	54	52	60	50	108	113
363	327	65	62	57	73	49	63	53
426	426	64	49	72	51	42	70	73
499	499	64	75	83	82	72	109	158
327	363	61	49	41	50	37	39	49
281	2006	60	50	34	32	27	64	70
363	363	48	38	47	61	47	81	66
327	327	45	46	50	77	67	81	81
318	477	41	44	41	25	39	4	3
329	147	39	42	38	35	39	21	14
2006	281	39	42	34	22	26	55	76
239	270	38	33	43	30	26	7	19
327	329	38	39	45	19	41	6	5
2006	499	37	53	25	40	32	46	59
318	524	36	34	37	25	22	1	2
327	427	36	38	46	19	23	36	24

According to the data from the table, the most frequently visited stations are: Central Park S & 6 Ave, Grand Army Plaza & Central Park S, and Centre St & Chambers St

2. Create appropriate SQL expressions to find permanently dormant or vacant stations.

   SQL for Dormant Stations

   ```
   SELECT DISTINCT end_station_id AS "Dormant Station",
   end_station_name AS "Dormant Station Name",
   end_station_latitude AS "Dormant Station Latitude",
   end_station_longitude AS "Dormant Station Longitude"
   FROM CitiBikeData
   WHERE end_station_id NOT IN (SELECT start_station_id FROM CitiBikeData);
   ```

   SQL for Vacant Stations

   ```
   SELECT DISTINCT start_station_id AS “Vacant Station”,
   start_station_name AS "Vacant Station Name",
   start_station_latitude AS "Vacant Station Latitude",
   start_station_longitude AS "Vacant Station Longitude"
   FROM CitiBikeData
   WHERE start_station_id NOT IN (SELECT end_station_id FROM CitiBikeData);
   ```

   Both of these queries do not return any rows because all the stations have been accounted for in this data set.

3. **Bonus**: Reconstruct the tables and recreate the SQL expressions above to include zip codes, hence allowing for aggregation by area. Note that zip codes are not included in the data provided but may be related to the StationId by its Latitude and Longitude. This could be part of your research and may require some coding!

[IV]

Submit a written report that includes:
1- The complete E/R diagram and schema of the relational database fully specifying the given requirements and any other requirements gathered. Identify all keys and foreign keys of the database relations.
2- SQL code that creates the tables' structure.
3- SQL code that loads the data.
4- SQL code that answers the given queries.
5- Sample outputs for your SQL code.
6- Provide a narrative explaining the outputs produced whenever possible.

SQL code importing CSV File

```
DROP DATABASE IF EXISTS CitiBike;
CREATE DATABASE CitiBike;
USE CitiBike;

/* Temporary Table Containing All Data*/
CREATE TABLE CitiBikeData(
        trip_duration INT,
        start_time TIMESTAMP,
         start_day VARCHAR(9),
        stop_time TIMESTAMP,
        stop_day VARCHAR(9),
        start_station_id INT,
        start_station_name VARCHAR(255),
        start_station_latitude DECIMAL(10,8),
        start_station_longitude DECIMAL(10,8),
        end_station_id INT,
        end_station_name VARCHAR(255),
        end_station_latitude DECIMAL(10,8),
        end_station_longitude DECIMAL(10,8),
        bike_id INT,
        usertype ENUM("Subscriber","Customer"),
        birth_year SMALLINT,
        gender INT
);

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/data_set.csv'
INTO TABLE CitiBikeData
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
```

```
IGNORE 1 ROWS
(trip_duration, @start_time, start_day, @stop_time, stop_day, start_station_id, start_station_name,
start_station_latitude, start_station_longitude, end_station_id, end_station_name,
end_station_latitude, end_station_longitude, bike_id, usertype, birth_year, gender)
SET
start_time = STR_TO_DATE(@start_time, "%m/%d/%Y %H:%i %r"),
stop_time = STR_TO_DATE(@stop_time, "%m/%d/%Y %H:%i %r"),
start_day = DAYNAME(start_time),
stop_day = DAYNAME(stop_time);
```

Sample Outputs

Stations

Id	Name	Latitude	Longitude
72	W 52 St & 11 Ave	40.76727216	-73.99392888
79	Franklin St & W Broadway	40.71911552	-74.00666661
82	St James Pl & Pearl St	40.71117416	-74.00016545
83	Atlantic Ave & Fort Greene Pl	40.68382604	-73.97632328
116	W 17 St & 8 Ave	40.74177603	-74.00149746
119	Park Ave & St Edwards St	40.69608941	-73.97803415
120	Lexington Ave & Classon Ave	40.68676793	-73.95928168
127	Barrow St & Hudson St	40.73172428	-74.00674436
128	MacDougal St & Prince St	40.72710258	-74.00297088
137	E 56 St & Madison Ave	40.76162800	-73.97292400
143	Clinton St & Joralemon St	40.69239502	-73.99337909
144	Nassau St & Navy St	40.69839895	-73.98068914
146	Hudson St & Reade St	40.71625008	-74.00910590
147	Greenwich St & Warren St	40.71542197	-74.01121978
150	E 2 St & Avenue C	40.72087360	-73.98085795
151	Cleveland Pl & Spring St	40.72181580	-73.99720307
152	Warren St & Church St	40.71473993	-74.00910627
153	E 40 St & 5 Ave	40.75206231	-73.98163240

Trips

StationId	MinTripDuration	MaxTripDuration	AvgTripDuration	NumberStartUsers	NumberReturnUsers
72	62	91003	1126	3575	3508
79	61	37668	1083	5034	5131
82	79	32300	1090	1141	1086
83	65	85639	1230	1589	1889
116	60	35062	797	3521	3388
119	72	2528	898	117	127
120	61	15150	1034	651	709
127	60	70573	960	4488	4244
128	60	46537	899	4638	4770
137	61	114650	1238	647	712
143	61	11425	990	916	912
144	60	18857	1235	417	370
146	61	32285	894	2368	2153
147	60	74233	936	4012	4222
150	69	82238	1029	1604	1733
151	61	6250750	2020	5655	5778
152	64	50607	1034	2654	2655
153	62	38159	845	3050	3025

UsageByDay

StationId	NumberWeekdayStartUsers	NumberWeekdayReturnUsers	NumberWeekendStartUsers	NumberWeekendReturnUsers
72	2684	891	2659	849
79	3653	1381	3727	1404
82	818	323	772	314
83	1105	484	1387	502
116	2811	710	2708	680
119	82	35	95	32
120	458	193	495	214
127	3607	881	3480	764
128	3487	1151	3602	1168
137	473	174	547	165
143	643	273	655	257
144	224	193	212	158
146	1863	505	1674	479
147	2931	1081	3026	1196
150	1188	416	1321	412
151	3963	1692	4090	1688
152	1997	657	1945	710
153	2470	580	2485	540

UsageByGender

StationId	NumberMaleStartUsers	NumberFemaleStartUsers	NumberMaleReturnUsers	NumberFemaleReturnUsers
72	2144	645	2166	606
79	2624	1076	2639	1022
82	592	203	555	194
83	820	305	1034	358
116	2350	676	2352	622
119	54	39	60	40
120	393	179	430	168
127	2731	957	2662	898
128	2854	974	2892	1045
137	358	90	378	100
143	557	211	590	189
144	140	53	125	54
146	1461	522	1356	464
147	2371	824	2556	904
150	1043	332	1099	344
151	3161	1165	3283	1095
152	1642	438	1576	499
153	2013	474	1915	533

UsageByAge

StationId	NumberMaleUsersUnder18	NumberMaleUsers18To40	NumberMaleUsersOver40	NumberFemaleUsersUnder18	NumberFemaleUsers18To40	NumberFemaleUsersOver40
72	6	1370	767	0	481	164
79	9	1668	947	5	776	295
82	0	387	205	0	99	104
83	1	597	222	0	234	71
116	2	1561	787	0	484	192
119	0	33	21	0	37	2
120	0	292	101	0	135	44
127	6	1723	1002	4	694	259
128	2	1779	1073	0	699	275
137	0	245	113	0	67	23
143	3	329	225	0	145	66
144	0	105	35	0	42	11
146	0	789	672	3	296	223
147	0	1520	851	2	567	255
150	0	788	255	0	252	80
151	0	2309	852	1	882	282
152	3	965	674	1	302	135
153	2	1180	831	0	331	143