22017160

Database Management.

Using Sqlite3 I started by establishing a database connection. Then a table was created named Carsharing using the Csv file provided.

1. import sqlite3

import pandas as pd

# Establish a DB connection

con = sqlite3.connect("Carsharing.db", isolation\_level=None)

cur = con.cursor()

# Create CarSharing Table

car\_sharing = pd.read\_csv("CarSharing.csv")

car\_sharing.to\_sql("CarSharing", con, if\_exists='replace', index=False)

A backup table was then further created adding the data into it. It included details like Timestamp, season, Holiday...

#Create Backup table and copy data

cur.execute(

"CREATE TABLE IF NOT EXISTS Backup (Id INTEGER NOT NULL, Timestamp INTEGER NOT NULL, Season TEXT NOT NULL, Holiday TEXT, Working\_Day TEXT, Weather TEXT, Temp REAL, Temp\_feel REAL, Humidity INTEGER, Windspeed REAL, Demand REAL NOT NULL)"

)

cur.execute("INSERT INTO Backup SELECT \* FROM CarSharing")

2. A new column called Temp\_category was created by altering the table, it included three strings for when the temperature felt like <10, <= 25,>25, respectively.

cur.execute("ALTER TABLE CarSharing ADD COLUMN temp\_category TEXT")

cur.execute("""

UPDATE CarSharing

SET temp\_category = CASE

WHEN Temp\_Feel < 10 THEN 'Cold'

WHEN Temp\_Feel <= 25 THEN 'Mild'

WHEN Temp\_Feel > 25 THEN 'Hot'

END

WHERE Temp\_Feel IS NOT NULL

""")

3. I then further created a new table called temperature which included columns from the carsharing table like temp, temp\_feel and the new column created temp\_category.

after which I dropped the two columns (temp, and temp\_feel from the carsharing table)

#Create temperature table and restructure columns in CarSharing

cur.executescript("""

BEGIN;

CREATE TABLE temperature AS SELECT temp, temp\_feel, temp\_category FROM CarSharing;

ALTER TABLE CarSharing DROP COLUMN temp;

ALTER TABLE CarSharing DROP COLUMN temp\_feel;

COMMIT;

""")

4. A new column was then added called weather\_code by altering the carsharing table. The weather column was then assigned values by updating the table, it included cases like for when the weather was clear, then 1 and so on...

#Add weather\_code column

cur.execute("ALTER TABLE CarSharing ADD COLUMN weather\_code TEXT")

cur.execute("""

UPDATE CarSharing

SET weather\_code = CASE

WHEN weather == 'Clear or partly cloudy' THEN 1

WHEN weather == 'Light snow or rain' THEN 2

WHEN weather == 'Mist' THEN 3

WHEN weather == 'heavy rain/ice pellets/snow + fog' THEN 4

END

WHERE weather IS NOT NULL

""")

5. I then created a new table called weather and it included the columns from the carsharing table called weather and weather\_code, this was done by selecting from the table.

The Carsharing table was then later restructured by droping the weather column from it.

#Create weather table and restructure columns in CarSharing

cur.executescript("""

BEGIN;

CREATE TABLE weather AS SELECT weather, weather\_code FROM CarSharing;

ALTER TABLE CarSharing DROP COLUMN weather;

COMMIT;

""")

6. I then created a new table called time that contained four columns with each row’s timestamp, hour,

weekday name, and month name. I did this by using the surf time () function.

#Create timetable

cur.execute(

"CREATE TABLE time AS SELECT timestamp, strftime('%H',timestamp) as 'hour', strftime('%w',timestamp) as 'weekday name', strftime('%m',timestamp) as 'month name' from CarSharing"

)

7a. As requested, I was to determine the date and time which had the highest demand rate in 2017, and I wrote a query considering the added details of the car sharing table.

#SELECT timestamp, max(demand) FROM CarSharing WHERE (timestamp BETWEEN "2017-01-01 00:00:00 " AND "2017-12-31 23:59:59 " )

# the query to fetch the date and time we had the highest demand rate in 2017.

cur.execute('''select max(demand) from CarSharing where timestamp like '%2017%' ;''')

print(cur.fetchone())

From the above, the date and time with the highest demand rate was 2017-06-15 and 17:00:00, respectively.

7b. Then I created a table containing the weekday, month and season showing the highest and lowest average demand rates for 2017 and the average demand values.

cur.execute('''create table seasonalavg as select

case cast (strftime('%w', timestamp) as integer)

when 0 then 'Sunday'

when 1 then 'Monday'

when 2 then 'Tuesday'

when 3 then 'Wednesday'

when 4 then 'Thursday'

when 5 then 'Friday'

else 'Saturday' end as weekdayname,

case cast (strftime('%m', date(timestamp)) as integer)

when '01' then 'January'

when '02' then 'February'

when '03' then 'March'

when '04' then 'April'

when '05' then 'May'

when '06' then 'June'

when '07' then 'July'

when '08' then 'August'

when '09' then 'September'

when '10' then 'October'

when '11' then 'November'

else 'December' end as month,season,demand, avg(demand) as demand\_average from CarSharing where timestamp like '%2017%'

group by weekdayname, month, season;''')

cur.execute('select weekdayname, month, season from seasonalavg where demand\_average = (select max(demand\_average) from seasonalavg) ;')

print(cur.fetchall())

cur.execute('select weekdayname, month, season from seasonalavg where demand\_average = (select min(demand\_average) from seasonalavg) ;')

print(cur.fetchall())

From the above the weekday, month and season with the highest average demand was “0” “07” “fall” with 0 being Sunday and 07 as July. Their calculated average demand rate was 4.997135078747038

The weekday month and season with lowest average demand rate was “1” “01” “spring” with 1 being Monday and 01 represented as January. The calculated average demand rate is 3.0507857781010803.

7c. Using the provided information from the wekdays, as requested I was to create a table showing the average demand rates at different hours from those days throughout 2017. I created two tables called Sunday\_demand and Monday\_demand.

cur.execute(''' create table sundays\_demand as select

strftime('%H',timestamp) as "Hour",

case cast (strftime('%w', timestamp) as integer)

when 0 then 'Sunday'

when 1 then 'Monday'

when 2 then 'Tuesday'

when 3 then 'Wednesday'

when 4 then 'Thursday'

when 5 then 'Friday'

else 'Saturday' end as weekday,

case cast (strftime('%m', date(timestamp)) as integer)

when '01' then 'January'

when '02' then 'February'

when '03' then 'March'

when '04' then 'April'

when '05' then 'May'

when '06' then 'June'

when '07' then 'July'

when '08' then 'August'

when '09' then 'September'

when '10' then 'October'

when '11' then 'November'

else 'December' end as monthname,season,demand from CarSharing where timestamp like '%2017%' and weekday = 'Sunday';''')

cur.execute(''' create table mondays\_demand as select

strftime('%H',timestamp) as "Hour",

case cast (strftime('%w', timestamp) as integer)

when 0 then 'Sunday'

when 1 then 'Monday'

when 2 then 'Tuesday'

when 3 then 'Wednesday'

when 4 then 'Thursday'

when 5 then 'Friday'

else 'Saturday' end as weekday,

case cast (strftime('%m', date(timestamp)) as integer)

when '01' then 'January'

when '02' then 'February'

when '03' then 'March'

when '04' then 'April'

when '05' then 'May'

when '06' then 'June'

when '07' then 'July'

when '08' then 'August'

when '09' then 'September'

when '10' then 'October'

when '11' then 'November'

else 'December' end as monthname,season,demand from CarSharing where timestamp like '%2017%' and weekday = 'Monday';''')

cur.execute('select \* from sundays\_demand;')

print(cur.fetchall())

cur.execute('select \* from mondays\_demand;')

print(cur.fetchall())

cur.execute('''select max(temp\_category) from CarSharing where timestamp like '%2017%' ;''')

print(cur.fetchall())

7d. The code below shows the weather conditions for the year 2017. There are two tables representing wind humidity. The queries are for the average, highest, and lowest wind speed, and humidity for each month in 2017.

cur.execute('''select max(weather) from CarSharing where timestamp like '%2017%' ;''')

print(cur.fetchall())

cur.execute('''select case cast (strftime('%m', date(timestamp)) as integer)

when '01' then 'January'

when '02' then 'February'

when '03' then 'March'

when '04' then 'April'

when '05' then 'May'

when '06' then 'June'

when '07' then 'July'

when '08' then 'August'

when '09' then 'September'

when '10' then 'October'

when '11' then 'November'

else 'December' end as monthname, avg(windspeed), max(windspeed), min(windspeed) from CARSHARING

where timestamp like '%2017%' group by monthname order by monthname;''')

print(cur.fetchall())

cur.execute('''select case cast (strftime('%m', date(timestamp)) as integer)

when '01' then 'January'

when '02' then 'February'

when '03' then 'March'

when '04' then 'April'

when '05' then 'May'

when '06' then 'June'

when '07' then 'July'

when '08' then 'August'

when '09' then 'September'

when '10' then 'October'

when '11' then 'November'

else 'December' end as monthname, avg(humidity), max(humidity), min(humidity) from CARSHARING

where timestamp like '%2017%' group by monthname order by monthname;''')

print(cur.fetchall())

7e. Finally, as requested we have the query showing the month with the highest average demand rate in 2017.

cur.execute('''select temp\_category, avg(demand) from CARSHARING where timestamp like '%2017%' group by temp\_category order by avg(demand) desc;''')

print(cur.fetchall())

cur.close()