SQL EDA

June 7, 2024

[1]:

```
##Assignment: Exploratory Data Analysis with SQL
install.packages("RSQLite")
library("RSQLite")
```

```
Updating HTML index of packages in '.Library'
Making 'packages.html' … done
```

[2]:

```
library(RJDBC)
```

```
Loading required package: DBI
Loading required package: rJava
```

[3]:

```
install.packages("RODBC")
library(RODBC)
```

```
Updating HTML index of packages in '.Library'
Making 'packages.html' … done
```

[4]:

```
#Enter the values for you database connection
dsn_driver = "com.ibm.db2.jcc.DB2Driver"
dsn_database = "bludb"             # e.g. "bludb"
dsn_hostname = "<54a2f15b-5c0f-46df-8954-.databases.appdomain.cloud>"   # e.g.␣
 ↪replace <yourhostname> with your hostname
dsn_port = ""                  # e.g. "3273"
dsn_protocol = "TCPIP"            # i.e. "TCPIP"
dsn_uid = "<zjh17769>"              # e.g. replace <username> with your userid
dsn_pwd = "<zcwd4+8gbq9bm5k4>"            # e.g. replace <password> with your␣
 ↪password
dsn_security <- "ssl"
```

[5]:

```
cc <- JDBC("com.ibm.db2.jcc.DB2Driver", "/home/jupyterlab/.rlang/db2jcc-db2jcc4.
 ↪jar")
jdbc_path <- paste("DRIVER=",dsn_driver,
                  ";DATABASE=",dsn_database,
                  ";HOSTNAME=",dsn_hostname,
                  ";PORT=",dsn_port,
                  ";PROTOCOL=",dsn_protocol,
                  ";UID=",dsn_uid,
                     ";PWD=",dsn_pwd,
                     ";SECURITY=",dsn_security,
                       sep="")
```

[6]:

```
#Connect to the database
conn <- dbConnect(RSQLite::SQLite(),"seoul_bike_sharing.sqlite")
```

[7]:

```
attributes(conn)
```

```
$ptr
<pointer: 0x5637c41174d0>

$dbname
[1] "seoul_bike_sharing.sqlite"

$loadable.extensions
[1] TRUE

$flags
[1] 70

$vfs
[1] ""

$ref
<environment: 0x5637c109d370>

$bigint
[1] "integer64"

$extended_types
[1] FALSE

$class
[1] "SQLiteConnection"
attr(,"package")
[1] "RSQLite"
```

[8]:

```
conn.info <- dbGetInfo(conn)
print(conn.info["db.version"])
print(conn.info["dbname"] )
```

```
$db.version
[1] "3.41.2"

$dbname
```

```
[1] "seoul_bike_sharing.sqlite"
```

[9]:

```
dbDataType(RSQLite::SQLite(), 1)
dbDataType(RSQLite::SQLite(), 1L)
dbDataType(RSQLite::SQLite(), "1")
dbDataType(RSQLite::SQLite(), TRUE)
dbDataType(RSQLite::SQLite(), list(raw(1)))

sapply(datasets::quakes, dbDataType, dbObj = RSQLite::SQLite())
```

'REAL'

'INTEGER'

'TEXT'

'INTEGER'

'BLOB'

lat 'REAL' **long** 'REAL' **depth** 'INTEGER' **mag** 'REAL' **stations** 'INTEGER'

[10]:

```
df1 <- dbExecute(conn,
                    "CREATE TABLE SEOUL_BIKE_SHARING (
                                  DATE NOT NULL,
                                  RENTED_BIKE_COUNT NOT NULL,
                                  HOUR NOT NULL,
                                  TEMPRETURE FLOAT(6),
                                  HUMIDITY  NOT NULL,
                                  WIND_SPEED FLOAT(6),
                                  VISIBILITY NOT NULL,
                                  PRESSURE INTEGER NOT NULL,
                                  DEW_POINT_TEMPRETURE FLOAT(6),
                                  SOLAR_RADIATION FLOAT(6),
                                  SEASON INTEGER VARCHAR(20) NOT NULL,
                                  HOLIDAY INTEGER VARCHAR(20) NOT NULL,
                                  FUNCTION_DAY INTEGER VARCHAR(20) NOT NULL
                                  )",
                    errors=FALSE
                    )

    if (df1 == -1){
        cat ("An error has occurred.\n")
        msg <- odbcGetErrMsg(conn)
        print (msg)
    } else {
        cat ("Table was created successfully.\n")
         }
```

```
Table was created successfully.
```

[11]:

```
df2 <- dbExecute(conn,
                    "CREATE TABLE CITIES_WEATHER_FORECAST (
                                       CITY INTEGER VARCHAR(20) NOT NULL,
                                       WEATHER INTEGER VARCHAR(20) NOT NULL,
                                       VISIBILTY NOT NULL,
                                       TEMP FLOAT(6),
                                       TEMP_MIN FLOAT(6),
                                       TEMP_MIX FLOAT(6),
                                       PRESSURE INTEGER VARCHAR(20) NOT NULL,
                                       HUMIDITY INTEGER VARCHAR(20) NOT NULL,
                                       WIND_SPEED FLOAT(6),
                                       WID_DEG INTEGER VARCHAR(20) NOT NULL,
                                       FORECAST_DATETIME NOT NULL
                                      )",
                    errors=FALSE
                    )

    if (df1 == -1){
        cat ("An error has occurred.\n")
        msg <- odbcGetErrMsg(conn)
        print (msg)
    } else {
        cat ("Table was created successfully.\n")
         }
```

```
Table was created successfully.
```

[12]:

```
df3 <- dbExecute(conn,
                    "CREATE TABLE BIKE_SHARING_SYSTEM (
                                       CITY INTEGER VARCHAR(20) NOT NULL,
                                       WEATHER INTEGER VARCHAR(20) NOT NULL,
                                       VISIBILTY NOT NULL,
                                       TEMP FLOAT(6),
                                       TEMP_MIN FLOAT(6),
                                       TEMP_MIX FLOAT(6),
                                       PRESSURE INTEGER VARCHAR(20) NOT NULL,
                                       HUMIDITY INTEGER VARCHAR(20) NOT NULL,
                                       WIND_SPEED FLOAT(6),
                                       WIND_DEG INTEGER VARCHAR(20) NOT NULL,
                                       FORECAST_DATETIME NOT NULL
                                      )",
                    errors=FALSE
                    )

    if (df1 == -1){
        cat ("An error has occurred.\n")
        msg <- odbcGetErrMsg(conn)
        print (msg)
    } else {
        cat ("Table was created successfully.\n")
         }
```

```
Table was created successfully.
```

[13]:

```
df4 <- dbExecute(conn,
                    "CREATE TABLE WORLD_CITIES (
                                      CITY INTEGER VARCHAR(20) NOT NULL,
                                      CITY_ASCII INTEGER VARCHAR(20) NOT NULL,
                                      LAT FLOAT(6),
                                      LING FLOAT(6),
                                      COUNTRY INTEGER VARCHAR(20) NOT NULL,
                                      ISO2 VARCHAR(20) NOT NULL,
                                      ISO3 VARCHAR(20) NOT NULL,
                                      ADMIN_NAME VARCHAR(20) NOT NULL,
                                      CAPITAL VARCHAR(20) NOT NULL,
                                      POPULATION NOT NULL,
                                      ID NOT NULL
                                      )",
                    errors=FALSE
                    )

    if (df1 == -1){
        cat ("An error has occurred.\n")
        msg <- odbcGetErrMsg(conn)
        print (msg)
    } else {
        cat ("Table was created successfully.\n")
    }
```

```
Table was created successfully.
```

[14]:

```
#check list of tables in the present db.
dbListTables(conn)
```

1. ’BIKE_SHARING_SYSTEM’ 2. ’CITIES_WEATHER_FORECAST’
3. ’SEOUL_BIKE_SHARING’ 4. ’WORLD_CITIES’

[16]:

```
df1 <- read.csv('seoul_bike_sharing.csv')
df2 <- read.csv('cities_weather_forecast.csv')
df3 <- read.csv('bike_sharing_systems.csv')
df4 <- read.csv('world_cities.csv')
head(df1)
head(df2)
head(df3)
head(df4)
```

A data.frame: 6 × 14

	DATE <fct>	RENTED_BIKE_COUNT <int>	HOUR <int>	TEMPERATURE <dbl>	HUMIDITY <int>
1	01/12/2017	254	0	-5.2	37
2	01/12/2017	204	1	-5.5	38
3	01/12/2017	173	2	-6.0	39
4	01/12/2017	107	3	-6.2	40
5	01/12/2017	78	4	-6.0	36
6	01/12/2017	100	5	-6.4	37

A data.frame: 6 × 12

	CITY <fct>	WEATHER <fct>	VISIBILITY <int>	TEMP <dbl>	TEMP_MIN <dbl>	TEMP_MAX <dbl>	PRESS <int>
1	Seoul	Clear	10000	12.32	10.91	12.32	1015
2	Seoul	Clear	10000	11.48	9.81	11.48	1016
3	Seoul	Clouds	10000	9.99	8.82	9.99	1015
4	Seoul	Clouds	10000	7.87	7.87	7.87	1014
5	Seoul	Clouds	10000	10.09	10.09	10.09	1014
6	Seoul	Rain	10000	9.74	9.74	9.74	1014

A data.frame: 6 × 4

	COUNTRY <fct>	CITY <fct>	SYSTEM <fct>	BICYCLES <int>
1	Albania	Tirana	NA	200
2	Argentina	Mendoza	NA	40
3	Argentina	San Lorenzo, Santa Fe	Biciudad	80
4	Argentina	Buenos Aires	Serttel Brasil	4000
5	Argentina	Rosario	NA	480
6	Australia	Melbourne	PBSC & 8D	676

A data.frame: 6 × 11

	CITY <fct>	CITY_ASCII <fct>	LAT <dbl>	LNG <dbl>	COUNTRY <fct>	ISO2 <fct>	ISO3 <fct>	ADMI <fct>
1	Tokyo	Tokyo	35.6897	139.6922	Japan	JP	JPN	Tōkyō
2	Jakarta	Jakarta	-6.2146	106.8451	Indonesia	ID	IDN	Jakart
3	Delhi	Delhi	28.6600	77.2300	India	IN	IND	Delhi
4	Mumbai	Mumbai	18.9667	72.8333	India	IN	IND	Mahār
5	Manila	Manila	14.5958	120.9772	Philippines	PH	PHL	Manila
6	Shanghai	Shanghai	31.1667	121.4667	China	CN	CHN	Shang

[17]:

```
dbWriteTable(conn, "SEOUL_BIKE_SHARING", df1, overwrite=TRUE, header = TRUE)
dbWriteTable(conn, "CITIES_WEATHER_FORECAST", df2, overwrite=TRUE, header =␣
 ↪TRUE)
dbWriteTable(conn, "BIKE_SHARING_SYSTEM", df3, overwrite=TRUE, header = TRUE)
dbWriteTable(conn, "WORLD_CITIES", df4, overwrite=TRUE, header = TRUE)
```

[18]:

```
##ask 1 - Record Count Determine how many records are in the seoul_bike_sharing␣
 ↪dataset.
dbGetQuery(conn, 'SELECT COUNT(DATE)
FROM SEOUL_BIKE_SHARING')
```

```
                      COUNT(DATE)
A data.frame: 1 × 1   <int>
                      8465
```

[19]:
```
#Task 2 - Operational Hours Determine how many hours had non-zero rented bike count
dbGetQuery(conn, 'SELECT COUNT(HOUR) COUNT_HOUR
FROM SEOUL_BIKE_SHARING
WHERE RENTED_BIKE_COUNT != 0')
```

```
                      COUNT_HOUR
A data.frame: 1 × 1   <int>
                      8465
```

[20]:
```
#Task 3 - Weather Outlook Query the the weather forecast for Seoul over the next 3 hours
dbGetQuery(conn,'SELECT *
FROM CITIES_WEATHER_FORECAST
ORDER BY FORECAST_DATETIME
LIMIT 1')
```

```
                       CITY    WEATHER   VISIBILITY   TEMP    TEMP_MIN   TEMP_MAX   PRESSUR
A data.frame: 1 × 12   <chr>   <chr>     <int>        <dbl>   <dbl>      <dbl>      <int>
                       Seoul   Clear     10000        12.32   10.91      12.32      1015
```

[21]:
```
#Task 4 - Seasons Find which seasons are included in the seoul bike sharing dataset
dbGetQuery(conn,'SELECT DISTINCT SEASONS
FROM SEOUL_BIKE_SHARING')
```

```
                      SEASONS
                      <chr>
                      Winter
A data.frame: 4 × 1   Spring
                      Summer
                      Autumn
```

[22]:
```
#Task 5 - Date Range Find the first and last dates in the Seoul Bike Sharing dataset
dbGetQuery(conn,'SELECT MAX(DATE) FIRST_DATE,
        MIN(DATE) LAST_DATE
FROM SEOUL_BIKE_SHARING')
```

```
                      FIRST_DATE   LAST_DATE
A data.frame: 1 × 2   <chr>        <chr>
                      31/12/2017   01/01/2018
```

[23]:

```
#Task 6 - Subquery - 'all-time high' determine which date and hour had the most␣
  ↪bike rentals
dbGetQuery(conn,"SELECT MAX(DATE) ,
                        MAX(HOUR)
FROM SEOUL_BIKE_SHARING ")
```

A data.frame: 1 × 2

MAX(DATE) <chr>	MAX(HOUR) <int>
31/12/2017	23

[24]:

```
#Task 7 - Hourly popularity and temperature by season determine the average␣
  ↪hourly temperature and
#the average number of bike rentals per hour over each season. List the top ten␣
  ↪results by average
#bike count.
dbGetQuery(conn,'SELECT SEASONS,
        HOUR,
        AVG(TEMPERATURE) AVG_TEMP,
        AVG(RENTED_BIKE_COUNT) AVG_RENTED_BIKE_COUNT
FROM SEOUL_BIKE_SHARING
GROUP BY SEASONS, HOUR
ORDER BY AVG_RENTED_BIKE_COUNT DESC
LIMIT 10')
```

A data.frame: 10 × 4

SEASONS <chr>	HOUR <int>	AVG_TEMP <dbl>	AVG_RENTED_BIKE_COUNT <dbl>
Summer	18	29.38791	2135.141
Autumn	18	16.03185	1983.333
Summer	19	28.27378	1889.250
Summer	20	27.06630	1801.924
Summer	21	26.27826	1754.065
Spring	18	15.97222	1689.311
Summer	22	25.69891	1567.870
Autumn	17	17.27778	1562.877
Summer	17	30.07691	1526.293
Autumn	19	15.06346	1515.568

[25]:

```
#Task 8 - Rental Seasonality Find the average hourly bike count during each␣
  ↪season
dbGetQuery(conn,'SELECT SEASONS,
        HOUR,
        AVG(RENTED_BIKE_COUNT) AVG_BIKE_COUNT,
        MIN(RENTED_BIKE_COUNT) MIN_BIKE_COUNT,
        MAX(RENTED_BIKE_COUNT) MAX_BIKE_COUNT
FROM SEOUL_BIKE_SHARING
GROUP BY HOUR, SEASONS
LIMIT 10')
```

A data.frame: 10 × 5

SEASONS <chr>	HOUR <int>	AVG_BIKE_COUNT <dbl>	MIN_BIKE_COUNT <int>	MAX_BIKE_CO <int>
Autumn	0	709.4375	119	1336
Spring	0	481.0889	22	1089
Summer	0	899.0652	26	1394
Winter	0	165.1778	42	342
Autumn	1	552.5000	144	1001
Spring	1	363.9444	23	837
Summer	1	698.7717	28	1088
Winter	1	159.0556	43	337
Autumn	2	377.4750	55	785
Spring	2	252.9667	9	590

[26]:

```
#Task 9 - Weather Seasonality Consider the weather over each season. On␣
 ↪average, what were the
#TEMPERATURE, HUMIDITY, WIND_SPEED, VISIBILITY,DEW_POINT_TEMPERATURE,␣
 ↪SOLAR_RADIATION,
#RAINFALL, and SNOWFALL per season?
dbGetQuery(conn,'SELECT SEASONS,
        AVG(TEMPERATURE) AVG_TEMP,
        AVG(HUMIDITY) AVG_HUMIDITY,
        AVG(WIND_SPEED) AVG_WIND_SPEED,
        AVG(VISIBILITY) AVG_VISIBILITY,
        AVG(DEW_POINT_TEMPERATURE) AVG_DEW_POINT_TEMP,
        AVG(SOLAR_RADIATION) AVG_SOLAR_RADIATION,
        AVG(RAINFALL) AVG_RAINFALL,
        AVG(SNOWFALL) AVG_SNOWFALL,
        AVG(RENTED_BIKE_COUNT) AVG_RENTED_BIKE_COUNT
FROM SEOUL_BIKE_SHARING
GROUP BY SEASONS
ORDER BY AVG_RENTED_BIKE_COUNT')
```

A data.frame: 4 × 10

SEASONS <chr>	AVG_TEMP <dbl>	AVG_HUMIDITY <dbl>	AVG_WIND_SPEED <dbl>	AVG_VISIBIL <dbl>
Winter	-2.540463	49.74491	1.922685	1445.987
Spring	13.021685	58.75833	1.857778	1240.912
Autumn	13.821580	59.04491	1.492101	1558.174
Summer	26.587711	64.98143	1.609420	1501.745

[27]:

```
#Task 10 - Total Bike Count and City Info for Seoul Use an implicit join across␣
 ↪the WORLD_CITIES
#and the BIKE_SHARING_SYSTEMS tables to determine the total number of bikes␣
 ↪avaialble in Seoul,
#plus the following city information about Seoul: CITY, COUNTRY, LAT, LON,␣
 ↪POPULATION, in a single view
dbGetQuery(conn, "SELECT CITY_ASCII, BS.COUNTRY, LAT, LNG, POPULATION, BICYCLES␣
 ↪NUM_BICYCLES
FROM WORLD_CITIES WC, BIKE_SHARING_SYSTEM BS
WHERE WC.CITY_ASCII = BS.CITY
AND
CITY_ASCII = 'Seoul';")
```

A data.frame: 1 × 6

CITY_ASCII <chr>	COUNTRY <chr>	LAT <dbl>	LNG <dbl>	POPULATION <dbl>	NUM_BICYCLES <int>
Seoul	South Korea	37.5833	127	21794000	20000

[28]:

```
#Task 11 - Find all city names and coordinates with comparable bike scale to␣
  ↪Seoul's bike sharing system
#Find all cities with total bike counts between 15000 and 20000. Return the␣
  ↪city and country names,
#plus the coordinates (LAT, LNG), population, and number of bicycles for each␣
  ↪city.
dbGetQuery(conn,"SELECT CITY_ASCII, BS.COUNTRY, LAT, LNG, POPULATION,␣
  ↪CAST(BICYCLES AS BIGINT)
FROM WORLD_CITIES WC, BIKE_SHARING_SYSTEM BS
WHERE WC.CITY_ASCII = BS.CITY
AND
BS.BICYCLES LIKE '15__' OR
BS.BICYCLES LIKE '16__' OR
BS.BICYCLES LIKE '17__' OR
BS.BICYCLES LIKE '18__' OR
BS.BICYCLES LIKE '19__' OR
BS.BICYCLES = '20000';")
```

A data.frame: 345414 × 6

CITY_ASCII	COUNTRY	LAT	LNG	POPULATION	CAST(BICYC
<chr>	<chr>	<dbl>	<dbl>	<dbl>	<int>
Tokyo	China	35.6897	139.6922	37977000	1600
Tokyo	China	35.6897	139.6922	37977000	20000
Tokyo	China	35.6897	139.6922	37977000	20000
Tokyo	China	35.6897	139.6922	37977000	20000
Tokyo	China	35.6897	139.6922	37977000	20000
Tokyo	Denmark	35.6897	139.6922	37977000	1860
Tokyo	France	35.6897	139.6922	37977000	1750
Tokyo	France	35.6897	139.6922	37977000	1852
Tokyo	Kazakhstan	35.6897	139.6922	37977000	1700
Tokyo	South Korea	35.6897	139.6922	37977000	20000
Tokyo	Taiwan	35.6897	139.6922	37977000	1695
Tokyo	United States	35.6897	139.6922	37977000	1833
Tokyo	United States	35.6897	139.6922	37977000	1800
Jakarta	China	-6.2146	106.8451	34540000	1600
Jakarta	China	-6.2146	106.8451	34540000	20000
Jakarta	China	-6.2146	106.8451	34540000	20000
Jakarta	China	-6.2146	106.8451	34540000	20000
Jakarta	China	-6.2146	106.8451	34540000	20000
Jakarta	Denmark	-6.2146	106.8451	34540000	1860
Jakarta	France	-6.2146	106.8451	34540000	1750
Jakarta	France	-6.2146	106.8451	34540000	1852
Jakarta	Kazakhstan	-6.2146	106.8451	34540000	1700
Jakarta	South Korea	-6.2146	106.8451	34540000	20000
Jakarta	Taiwan	-6.2146	106.8451	34540000	1695
Jakarta	United States	-6.2146	106.8451	34540000	1833
Jakarta	United States	-6.2146	106.8451	34540000	1800
Delhi	China	28.6600	77.2300	29617000	1600
Delhi	China	28.6600	77.2300	29617000	20000
Delhi	China	28.6600	77.2300	29617000	20000
Delhi	China	28.6600	77.2300	29617000	20000
Cheremoshna	South Korea	51.3894	30.0989	0	20000
Cheremoshna	Taiwan	51.3894	30.0989	0	1695
Cheremoshna	United States	51.3894	30.0989	0	1833
Cheremoshna	United States	51.3894	30.0989	0	1800
Ambarchik	China	69.6510	162.3336	0	1600
Ambarchik	China	69.6510	162.3336	0	20000
Ambarchik	China	69.6510	162.3336	0	20000
Ambarchik	China	69.6510	162.3336	0	20000
Ambarchik	China	69.6510	162.3336	0	20000
Ambarchik	Denmark	69.6510	162.3336	0	1860
Ambarchik	France	69.6510	162.3336	0	1750
Ambarchik	France	69.6510	162.3336	0	1852
Ambarchik	Kazakhstan	69.6510	162.3336	0	1700
Ambarchik	South Korea	69.6510	162.3336	0	20000
Ambarchik	Taiwan	69.6510	162.3336	0	1695
Ambarchik	United States	69.6510	162.3336	0	1833
Ambarchik	United States	69.6510	162.3336	0	1800
Nordvik	China	74.0165	111.5100	0	1600
Nordvik	China	74.0165	111.5100	0	20000
Nordvik	China	74.0165	111.5100	0	20000

[29]:

```
close(conn)
```

```
Error in UseMethod("close"): no applicable method for 'close' applied to an object of class "c('SQLiteConnection', 'DBIConnection', 'DBIObject')"
Traceback:

1. close(conn)
```

[ ]: