Particle Pollution and Respiratory Effects

Xu Chen

Introduction

Particulate Matter 2.5 is defined as tiny particles in the air that are two- and one-half microns or less in width, which can travel deeply into the respiratory tract, reaching the lungs (Department of Health, 2018). Scientific studies have linked increases in daily PM2.5 exposure with increased respiratory hospital admissions, emergency department visits and deaths (Department of Health, 2018).

Research Question

The goal of the study is to link the data about PM 2.5 levels to EHR data to find whether there is an association between PM2.5 levels and respiratory diseases. And the hypothesis I made is that increases in PM2.5 levels are associated with increased respiratory diseases.

Database Information

The data are from the Air Quality Measures on the National Environmental Health Tracking Network database provided by Centers for Disease Control and Prevention (CDC, 2018). The Environmental Protection Agency (EPA) provides air pollution data from the Air Quality System (AQS) about ozone and particulate matter (PM2.5) to CDC (CDC, 2018). Data from the AQS is considered the "gold standard" for determining outdoor air pollution (CDC, 2018). Table 1 is the snapshot of the database. One thing to be noted is that variable *MeasureType* has two value where 1 stands for Percent of days with PM2.5 levels over the National Ambient Air Quality Standard, and 2 stands for Annual average ambient concentrations of PM2.5 in micrograms per cubic meter.

Table 1. Air Quality Measures Table

	Record_id	FIPScode	StateName	CountyName	ReportYear	MeasureValue	MeasureType
1	51890	1001	Alabama	Autauga	2001	13.18680693	2
2	51891	1001	Alabama	Autauga	2002	12.4906683	2
3	51892	1001	Alabama	Autauga	2003	12.86948061	2
4	51893	1001	Alabama	Autauga	2004	13.11194507	2
5	51894	1001	Alabama	Autauga	2005	13.5432759	2
6	51895	1001	Alabama	Autauga	2006	13.63825239	2
7	51896	1001	Alabama	Autauga	2007	14.54288792	2
8	51897	1001	Alabama	Autauga	2008	12.50730434	2

ER Diagrams

ER Diagrams are shown in figure 1. The primary key and foreign keys are labeled in every table. The table *Air_quality* can be linked with table *Person* by *FIPScode*. The table *Person* can be linked with the table *Visit_occurrence*, the table *Condition_occurrence*, the table *Procedure_occurrence*, and the table *Drug_exposure* by *Person_id*. The table *Visit_occurrence* can be linked with the table *Condition_occurrence*, the table *Procedure_occurrence*, and the table *Drug_exposure* by *visit_occurrence_id*.

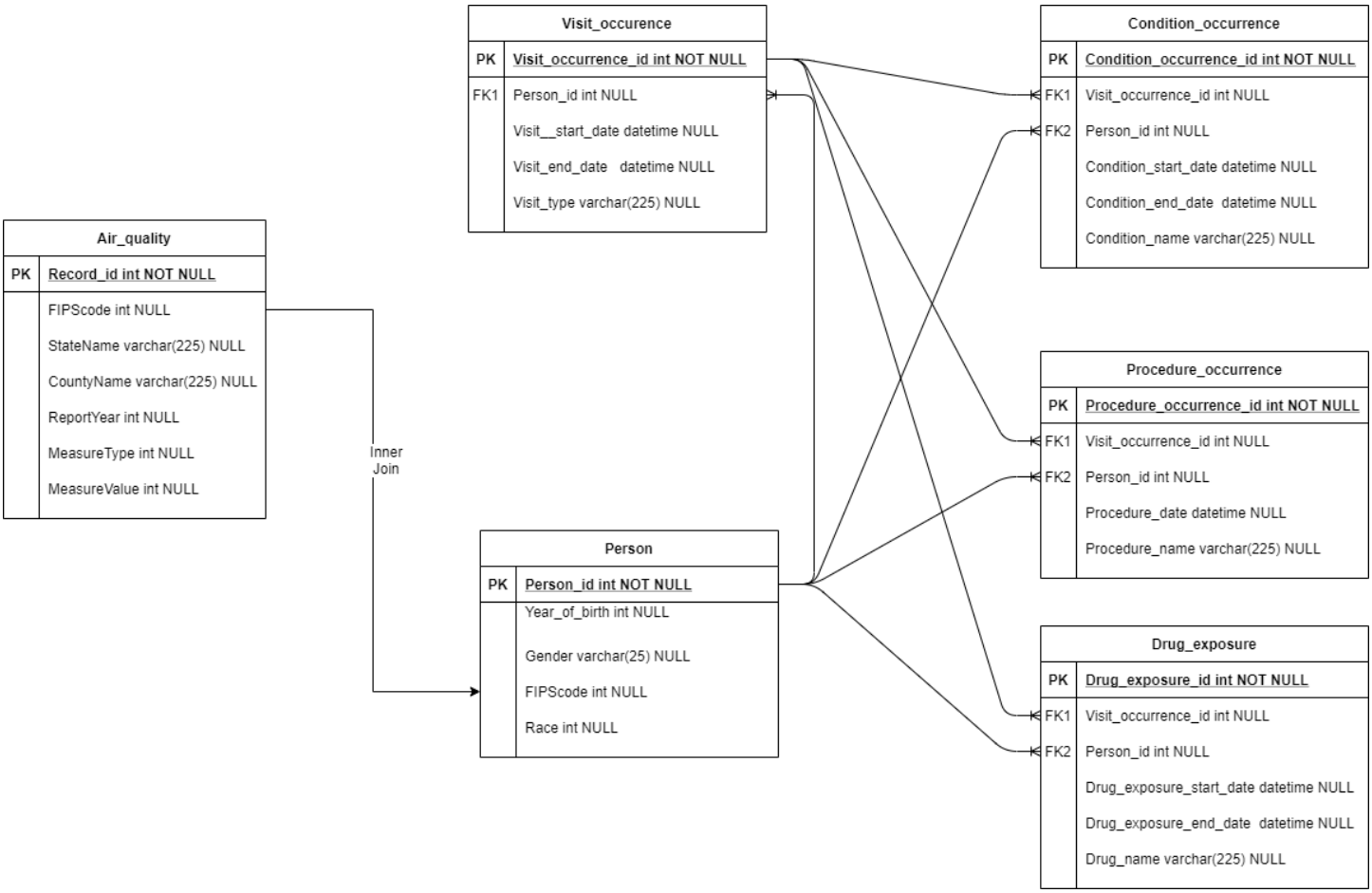


Figure 1. ER Diagrams of the SQL Server

Table Schema

Figure 2 shows the table schema. Primary key is placed first and underlined in every table,

following with all foreign keys and then other non-key attributes. The details of creating table will be shown in Appendix 1.

Figure 2. Table Schema

Air_Quality(Record_id, FIPS code, StateName, CountyName,
ReportYear, MeasureType, MeasureValue)
Person(Person_id, FIPS code, Year_of_birth, Gender, Race)
Visit_occurrence(Visit_occurrence, Person_id, Visit_start_date,
Visit_end_date, Visit_type)
Condition_occurrence(Condition_occurrence_id, Person_id,
Visit_occurrence_id, Condition_start_date, Condition_end_date, Condition_name)
Drug_exposure(Drug_exposure_id, Person_id, Visit_occurrence_id
Drug_exposure_start_date, Drug_ exposure_end_date, Drug_name)
Procedure_occurrence(Procedure_occurrence_id, Person_id,
Visit_occurrence_id, Procedure_date, Procedure_name)

Data dictionary

Data dictionary is shown in table 2, containing table name, filed name, data type, field length, constraint and description.

Table 2. Data dictionary

Table Name	Field Name	Data Type	Field Length	Constraint	Description
Air Quality	Record_id	int	10	Primary key	Record id, Auto generated
	FIPScode	int	10		Federal Information Processing Standards
	StateName	varchar	225		State name
	CountyName	varchar	225		County name
	ReportYear	int	10		The year when the data was measured
	MeasureType	int	10		Measure Type: 1 stands for Percent of days with PM2.5 levels over the National Ambient Air Quality Standard 2 stands for Annual average ambient concentrations of PM2.5 in micrograms per cubic meter
	MeasureValue	int	10		Measure value
Person	Person_id	int	10	Primary key	Person id, unique
	Year_of_birth	int	10		Year of birth
	Gender	varchar	25		Gender
	FIPScode	int	10		Federal Information Processing Standards
	Race	int	10		Race
Visit_occurence	Visit_occurrence_id	int	10	Primary key	Visit occurrence id, unique
	Person_id	int	10	Foreign key	References person_id of table Person
	Visit_start_date	datetime	8		Visit start date
	Visit_end_date	datetime	8		Visit end date
	Visit_type	var	225		Visit type: ED, inpatient, or outpatient
Condition_occurence	Condition_occurrence_id	int	10	Primary key	Condition occurrence id, unique
	Person_id	int	10	Foreign key	References person_id of table Person
	Visit_occurrence_id	int	10	Foreign key	References Visit_occurrence_id of table Visit_occurence
	Condition_start_date	datetime	8		Condition start date
	Condition_end_date	datetime	8		Condition end date
	Condition_name	varchar	225		Condition name
Drug_exposure	Drug_exposure_id	int	10	Primary key	Drug exposure id, unique
	Person_id	int	10	Foreign key	References person_id of table Person
	Visit_occurrence_id	int	10	Foreign key	References Visit_occurrence_id of table Visit_occurence
	Drug_exposure_start_date	datetime	8		Drug exposure start date
	Drug_exposure_end_date	datetime	8		Drug exposure end date
	Drug_name	varchar	225		Drug name
Procedure_occurrence	Procedure_occurrence_id	int	10	Primary key	Procedure occurrence id, unique
	Person_id	int	10	Foreign key	References person_id of table Person
	Visit_occurrence_id	int	10	Foreign key	References Visit_occurrence_id of table Visit_occurence
	Procedure_date	datetime	8		Procedure date
	Procedure_name	varchar	225		Procedure name

SQL Queries for Research

First, we want to link the table *Air_quality* with table *Person* on *FIPScode*.

QryPatientFIPSView=Person+AirQuality

SQL query:

```
CREATE VIEW QryPatientFIPSView AS

SELECT p.Person_id AS Person_id, a.*

FROM Person p

INNER JOIN

Air_quality a

ON p.FIPScode=a.FIPScode;


SELECT * FROM QryPatientFIPSView

ORDER BY person_id, ReportYear;
```

Output:

	Person_id	Record_id	FIPScode	StateName	CountyName	ReportYear	MeasureValue	MeasureType
1	1001	6607	1027	Alabama	Clay	1999	9.574468085	1
2	1001	9824	1027	Alabama	Clay	1999	17.40599817	2
3	1001	15218	1027	Alabama	Clay	2000	16.41977151	2
4	1001	8356	1027	Alabama	Clay	2000	3.80952381	1
5	1001	52033	1027	Alabama	Clay	2001	13.25673364	2
6	1001	17872	1027	Alabama	Clay	2001	0	1
7	1001	16444	1027	Alabama	Clay	2001	12.8168206	2
8	1001	1625	1027	Alabama	Clay	2001	0	1
9	1001	17873	1027	Alabama	Clay	2002	0.547945205	1
10	1001	52034	1027	Alabama	Clay	2002	12.49863226	2
11	1001	8532	1027	Alabama	Clay	2002	0.847457627	1
12	1001	13420	1027	Alabama	Clay	2002	13.18301724	2
13	1001	14758	1027	Alabama	Clay	2003	13.36892145	2
14	1001	52035	1027	Alabama	Clay	2003	12.75173043	2

Then, we want to link the table *Condition_occurrence* with the view QryPatientFIPSView.

QryConditionPatientFIPSView= Person+AirQuality+Condition_occurrence

SQL query:

```
CREATE VIEW QryConditionPatientFIPSView AS
SELECT c.*,q.FIPScode,StateName,CountyName,
ReportYear,MeasureValue,MeasureType
FROM Condition_occurrence c
INNER JOIN
QryPatientFIPSView q
ON c.Person_id=q.Person_id;

SELECT * FROM QryConditionPatientFIPSView;
```

Output:

	Condition_occurrence_id	Person_id	Condition_start_date	Condition_end_date	Condition_name	FIPScode	StateName	CountyName	ReportYear	MeasureValue	MeasureType
1	3001	1001	2000-01-01 00:00:00.000	2000-01-10 00:00:00.000	asthma	1027	Alabama	Clay	2006	0	1
2	3001	1001	2000-01-01 00:00:00.000	2000-01-10 00:00:00.000	asthma	1027	Alabama	Clay	1999	9.574468085	1
3	3001	1001	2000-01-01 00:00:00.000	2000-01-10 00:00:00.000	asthma	1027	Alabama	Clay	2007	2.5	1
4	3001	1001	2000-01-01 00:00:00.000	2000-01-10 00:00:00.000	asthma	1027	Alabama	Clay	2005	2.5	1
5	3001	1001	2000-01-01 00:00:00.000	2000-01-10 00:00:00.000	asthma	1027	Alabama	Clay	2008	0	1
6	3001	1001	2000-01-01 00:00:00.000	2000-01-10 00:00:00.000	asthma	1027	Alabama	Clay	2011	0	1
7	3001	1001	2000-01-01 00:00:00.000	2000-01-10 00:00:00.000	asthma	1027	Alabama	Clay	2000	3.80952381	1
8	3001	1001	2000-01-01 00:00:00.000	2000-01-10 00:00:00.000	asthma	1027	Alabama	Clay	2002	0.847457627	1
9	3001	1001	2000-01-01 00:00:00.000	2000-01-10 00:00:00.000	asthma	1027	Alabama	Clay	2013	0	1
10	3001	1001	2000-01-01 00:00:00.000	2000-01-10 00:00:00.000	asthma	1027	Alabama	Clay	1999	17.40599817	2
11	3001	1001	2000-01-01 00:00:00.000	2000-01-10 00:00:00.000	asthma	1027	Alabama	Clay	2010	10.51590608	2

Next step is using the WHERE clause to filter query results. We need a list of records where Report Year from *Air_Quality* is between the *condition_start_date* and the *condition_end_date* and patient's condition name is "asthma".

SQL query:

```
CREATE VIEW QryAsthmaConditionView AS
SELECT *  FROM QryConditionPatientFIPSView
WHERE ReportYear<=YEAR(Condition_end_date)
AND ReportYear>=YEAR(Condition_start_date)
AND Condition_name='asthma';

SELECT * FROM QryAsthmaConditionView;
```

Output:

	Condition_occurrence_id	Person_id	Condition_start_date	Condition_end_date	Condition_name	FIPScode	StateName	CountyName	ReportYear	MeasureValue	MeasureType
1	3001	1001	2000-01-01 00:00:00.000	2000-01-10 00:00:00.000	asthma	1027	Alabama	Clay	2000	3.80952381	1
2	3001	1001	2000-01-01 00:00:00.000	2000-01-10 00:00:00.000	asthma	1027	Alabama	Clay	2000	16.41977151	2
3	3003	1002	2001-01-01 00:00:00.000	2002-01-10 00:00:00.000	asthma	1051	Alabama	Elmore	2001	13.36508564	2
4	3004	1002	1999-01-01 00:00:00.000	2003-01-10 00:00:00.000	asthma	1051	Alabama	Elmore	2001	13.36508564	2
5	3003	1002	2001-01-01 00:00:00.000	2002-01-10 00:00:00.000	asthma	1051	Alabama	Elmore	2002	12.67708774	2
6	3004	1002	1999-01-01 00:00:00.000	2003-01-10 00:00:00.000	asthma	1051	Alabama	Elmore	2002	12.67708774	2
7	3004	1002	1999-01-01 00:00:00.000	2003-01-10 00:00:00.000	asthma	1051	Alabama	Elmore	2003	12.92985513	2
8	3003	1002	2001-01-01 00:00:00.000	2002-01-10 00:00:00.000	asthma	1051	Alabama	Elmore	2001	0.273972603	1
9	3004	1002	1999-01-01 00:00:00.000	2003-01-10 00:00:00.000	asthma	1051	Alabama	Elmore	2001	0.273972603	1
10	3003	1002	2001-01-01 00:00:00.000	2002-01-10 00:00:00.000	asthma	1051	Alabama	Elmore	2002	0	1

Finally, we would like to create tables of the time trends of PM2.5 data and the number of respiratory diseases. The first table shows the time trends of percent of days with PM2.5 levels over the National Ambient Air Quality Standard and the number of asthmatic patients in that year.

SQL query:

```
CREATE VIEW QryAsthmaType1View as
SELECT FIPScode,ReportYear,
AVG(MeasureValue) AS avg_MeasureValue,
COUNT(Condition_occurrence_id) AS Condition_Number
FROM QryAsthmaConditionView
WHERE MeasureType=1
GROUP BY ReportYear,FIPScode;
```

Output:

	FIPScode	ReportYear	avg_MeasureValue	Condition_Number
1	1027	2000	3.80952381	2
2	1027	2001	0	2
3	1027	2002	0.697701416	2
4	1027	2003	0.724423053	2
5	1027	2004	0.5464480875	2
6	1027	2005	1.7979452055	2
7	1027	2006	0	2
8	1051	2001	0.273972603	3
9	1051	2002	0	3
10	1051	2003	0.273972603	2
11	1051	2004	1.366120219	1
12	1051	2005	1.369863014	1
13	1051	2006	0	1
14	4003	2000	3.50877193	2
15	4003	2001	0	4
16	4003	2002	0	4

The second table shows the time trends of annual average ambient concentrations of PM2.5 in micrograms per cubic meter and the number of asthmatic patients in that year.

SQL query:

```
CREATE VIEW QryAsthmaType1View as
SELECT FIPScode,ReportYear,
AVG(MeasureValue) AS avg_MeasureValue,
COUNT(Condition_occurrence_id) AS Condition_Number
FROM QryAsthmaConditionView
WHERE MeasureType=2
GROUP BY ReportYear,FIPScode;
```

Output:

	FIPScode	ReportYear	avg_MeasureValue	Condition_Number
1	1027	2000	16.41977151	2
2	1027	2001	13.03677712	2
3	1027	2002	12.84082475	2
4	1027	2003	13.06032594	2
5	1027	2004	13.071755485	2
6	1027	2005	13.4243568	2
7	1027	2006	13.359181135	2
8	1051	2001	13.36508564	3
9	1051	2002	12.67708774	3
10	1051	2003	12.92985513	2
11	1051	2004	13.37454129	1
12	1051	2005	13.67482979	1
13	1051	2006	13.95563655	1
14	4003	2000	8.860285027	2
15	4003	2001	7.20964663	2
16	4003	2002	7.2423912665	4

We are also interested in the relative frequency of condition number.

SQL query:

```
SELECT A.FIPScode, A.ReportYear,A.avg_MeasureValue, ROUND(A.count1*1.0/B.count2,3) as Frequency
FROM (
SELECT FIPScode,ReportYear,avg(MeasureValue) as avg_MeasureValue,
COUNT(Condition_occurrence_id) AS count1
FROM QryAsthmaConditionView
WHERE MeasureType=1
GROUP BY ReportYear,FIPScode
) as A
INNER JOIN(
SELECT FIPScode,
COUNT(Condition_occurrence_id) AS count2
FROM QryAsthmaConditionView
WHERE MeasureType=1
GROUP BY FIPScode
) as B
ON A.FIPScode=B.FIPScode;
```

Output:

	FIPScode	ReportYear	avg_MeasureValue	Frequency
1	1027	2000	3.80952381	0.143000000000
2	1027	2001	0	0.143000000000
3	1027	2002	0.697701416	0.143000000000
4	1027	2003	0.724423053	0.143000000000
5	1027	2004	0.5464480875	0.143000000000
6	1027	2005	1.7979452055	0.143000000000
7	1027	2006	0	0.143000000000
8	1051	2001	0.273972603	0.273000000000
9	1051	2002	0	0.273000000000
10	1051	2003	0.273972603	0.182000000000
11	1051	2004	1.366120219	0.091000000000
12	1051	2005	1.369863014	0.091000000000
13	1051	2006	0	0.091000000000
14	4003	2000	3.50877193	0.071000000000
15	4003	2001	0	0.143000000000
16	4003	2002	0	0.143000000000
17	4003	2003	0	0.214000000000
18	4003	2004	0	0.214000000000

Visualization

ggplot2 is an excellent data visualization package for the statistical programming language R. Therefore, we are interested in connecting R to SQL Server. A special designed R-package called *RODBC* can help with this. The function *odbcConnect* can achieve connection and the function *sqlQuery* can submit an SQL query to an ODBC database, and retrieve the results (RDocumentation, 2020). R codes as below selected QryAsthmaType2View from the SQL Server as a database.

R codes:

```
library(RODBC)
myConn<-odbcConnect('localsql')
QryAsthmaType2View <-sqlQuery(myConn,"select * from QryAsthmaType2View")
head(QryAsthmaType2View)
```

R output:

	ReportYear <int>	FIPScode <int>	avg_MeasureValue <dbl>	Condition_Number <int>
1	2000	1027	3.8095238	2
2	2001	1027	0.0000000	2
3	2002	1027	0.6977014	2
4	2003	1027	0.7244231	2
5	2004	1027	0.5464481	2
6	2005	1027	1.7979452	2

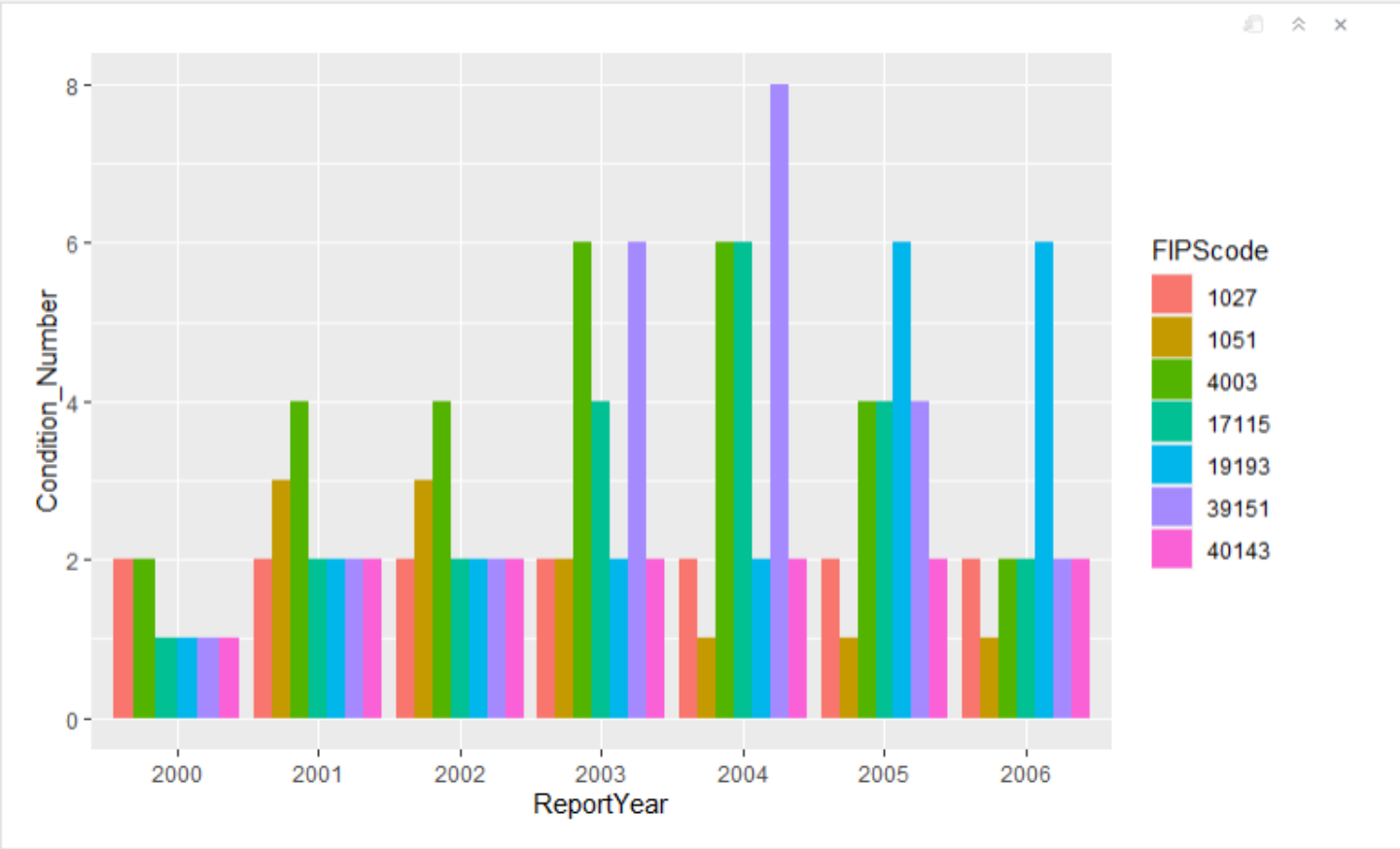
6 rows

Then using *ggplot2* package, we can plot the histogram of condition number as well as time trends of measure value (in this case, annual average ambient concentrations of PM2.5 in micrograms per cubic meter)

R codes:

```
library(ggplot2)
ggplot(data=QryAsthmaType2View)+
geom_line(aes(x=ReportYear,y=avg_MeasureValue,group=FIPScode,color=FIPScode),lwd=1)
```

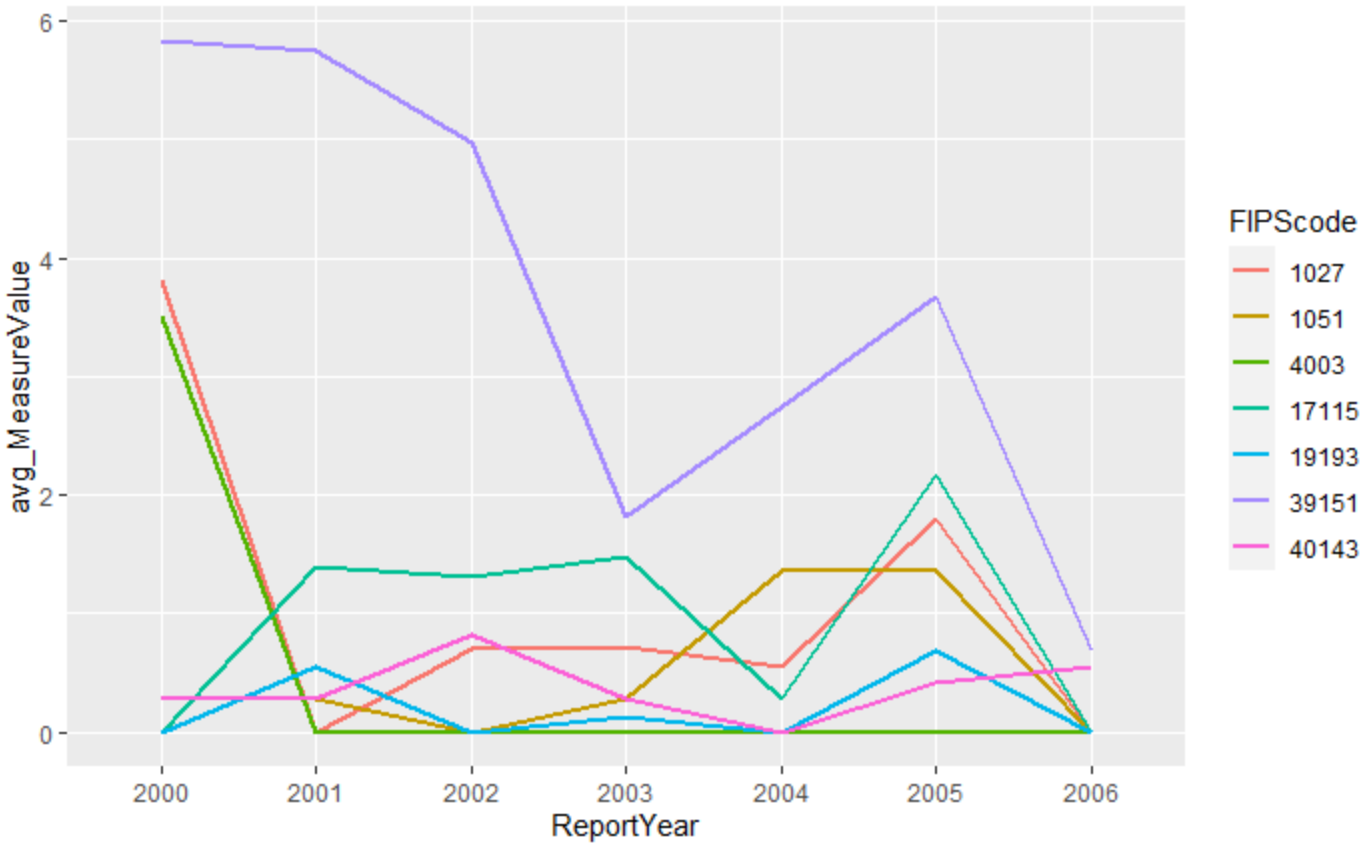
Figure 2. Histogram of Condition Number



R codes:

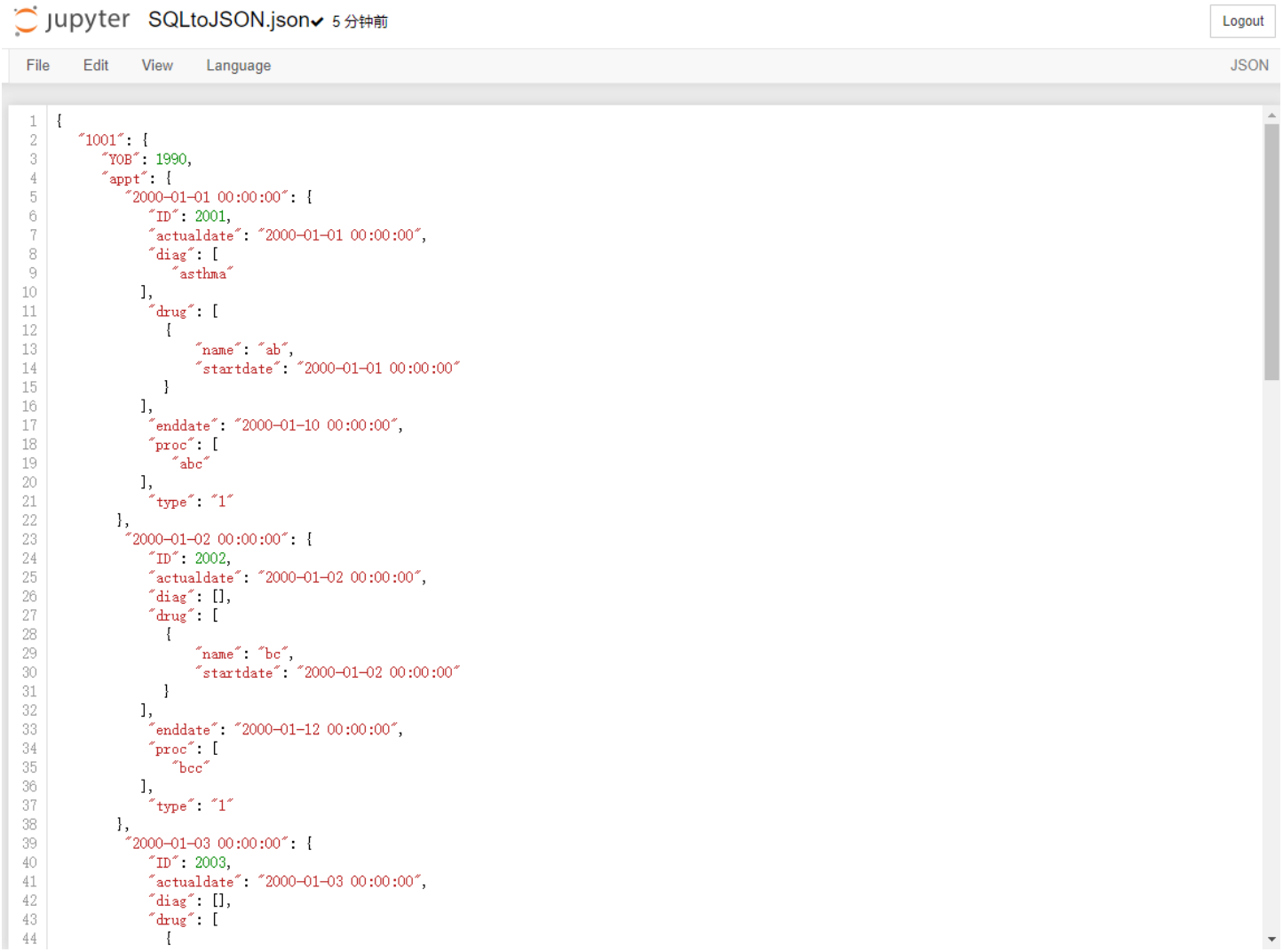
```
library(ggplot2)

ggplot(data=QryAsthmaType2View )+

geom_line(aes(x=ReportYear, y=avg_MeasureValue,group=FIPScode,color=FIPScode),lwd=1)
```

Figure 3. Time Trend of Measure Value



SQL to JSON

JSON file is shown as below. The code is in the Appendix 2, modified from “SQLtoJSONclass10.ipynb”.




```
{
  "1001": {
    "YOB": 1990,
    "appt": {
      "2000-01-01 00:00:00": {
        "ID": 2001,
        "actualdate": "2000-01-01 00:00:00",
        "diag": [
          "asthma"
        ],
        "drug": [
          {
            "name": "ab",
            "startdate": "2000-01-01 00:00:00"
          }
        ],
        "enddate": "2000-01-10 00:00:00",
        "proc": [
          "abc"
        ],
        "type": "1"
      },
      "2000-01-02 00:00:00": {
        "ID": 2002,
        "actualdate": "2000-01-02 00:00:00",
        "diag": [],
        "drug": [
          {
            "name": "bc",
            "startdate": "2000-01-02 00:00:00"
          }
        ],
        "enddate": "2000-01-12 00:00:00",
        "proc": [
          "bcc"
        ],
        "type": "1"
      },
      "2000-01-03 00:00:00": {
        "ID": 2003,
        "actualdate": "2000-01-03 00:00:00",
        "diag": [],
        "drug": [
          {
```

Reference:

1. Department of Health. (2018). Fine Particles (PM 2.5) Questions and Answers.

   Retrieved July 27, 2020, from https://www.health.ny.gov/environmental/indoors/air/pmq_a.htm#:~:text=Fine%20particulate%20matter%20(PM2.5,hazy%20when%20levels%20are%20elevated.

2. Centers for Disease Control and Prevention. (2018). the Air Quality Measures on the National

   Environmental Health Tracking Network. Retrieved July 27, 2020, from https://data.cdc.gov/Environmental-Health-Toxicology/Air-Quality-Measures-on-the-National-Environmental/cjae-szjv

3. RDocumentation. (2020). RODBC v1.3-17.

   Retrieved July 27, 2020, from

   https://www.rdocumentation.org/packages/RODBC/versions/1.3-17

Appendix 1 Create Table and Insert Data

1. Create Person table

```
Create table person(
Person_id int NOT NULL,
Year_of_birth int NULL,
Gender varchar(25) NULL,
FIPScode int NULL,
Race int NULL,
CONSTRAINT PK_Person PRIMARY KEY (Person_id)

);
```

Insert some synthetic data

```
INSERT into person VALUES ( '1001 ', '1990', 'Male', '1027', '1' );
INSERT into person VALUES ( '1002 ', '1980', 'Female', '1051', '1');
INSERT into person VALUES ( ' 1003 ', '1970', 'Male', '40143','1');
INSERT into person VALUES ( ' 1004 ', '1980', 'Female', '39151','1');
INSERT into person VALUES ( ' 1005 ', '1984', 'Male', '4003','1');
INSERT into person VALUES ( ' 1006 ', '1988', 'Male', '17115','1');
INSERT into person VALUES ( ' 1007 ', '1977', 'Female', '19193','1');
```

2. Create Visit_occurrence table

```
Create table Visit_occurrence (
Visit_occurrence_id int NOT NULL,
Person_id int NULL,
Visit_start_date datetime NULL,
Visit_end_date datetime NULL,
Visit_type varchar(225) NULL,
CONSTRAINT PK_Visit PRIMARY KEY (Visit_occurrence_id),
CONSTRAINT FK_Visit FOREIGN KEY (Person_id)
        REFERENCES Person(Person_id) ON UPDATE
CASCADE
```

```
);
```

Insert some synthetic data

```
INSERT into Visit_occurrence values( '2001 ', '1001', '2000-1-1 00:00:00', '2000-1-10 00:00:00','1');
INSERT into Visit_occurrence values( '2002 ', '1001', '2000-1-2 00:00:00', '2000-1-12 00:00:00','1');
INSERT into Visit_occurrence values( '2003 ', '1001', '2000-1-3 00:00:00', '2000-1-13 00:00:00','2');
INSERT into Visit_occurrence values( '2004 ', '1002', '2000-1-4 00:00:00', '2000-1-14 00:00:00','2');
INSERT into Visit_occurrence values( '2005 ', '1002', '2000-1-5 00:00:00', '2000-1-15 00:00:00','2');
```

3. Create Condition_occurrence table

```
Create table Condition_occurrence (
Condition_occurrence_id int NOT NULL,
Person_id int NULL,
Visit_occurrence_id int NULL,
Condition_start_date datetime NULL,
Condition_end_date datetime NULL,
Condition_name varchar(225) NULL,
CONSTRAINT PK_Con PRIMARY KEY (Condition_occurrence_id),
CONSTRAINT FK_Con FOREIGN KEY (Person_id)
        REFERENCES Person(Person_id),
CONSTRAINT FK_Con_Visit FOREIGN KEY (Visit_occurrence_id)
REFERENCES Visit_occurrence(Visit_occurrence_id) ON UPDATE
CASCADE
);
```

Insert synthetic data (~5 rows or more)

```
INSERT into Condition_occurrence values('3001', '1001', '2001','2000-1-1 00:00:00', '2000-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3002', '1002', '2002','1999-1-1 00:00:00', '2000-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3003', '1002', '2003','2001-1-1 00:00:00', '2002-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3004', '1002', '2004','1999-1-1 00:00:00', '2003-1-10 00:00:00',
'asthma');
```

```
INSERT into Condition_occurrence values('3005', '1004', '2005','2004-1-1 00:00:00', '2004-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3006', '1004', '2001','2005-1-1 00:00:00', '2005-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3007', '1004', '2002','1999-1-1 00:00:00', '2003-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3008', '1005', '2003','2004-1-1 00:00:00', '2005-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3009', '1005', '2004','2003-1-1 00:00:00', '2004-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3010', '1007', '2005','2003-1-1 00:00:00', '2004-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3011', '1006', '2001','2000-1-1 00:00:00', '2005-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3012', '1006', '2002','2003-1-1 00:00:00', '2004-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3013', '1007', '2003','2005-1-1 00:00:00', '2006-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3014', '1001', '2004','2004-1-1 00:00:00', '2005-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3015', '1002', '2005','2003-1-1 00:00:00', '2004-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3016', '1003', '2001','2005-1-1 00:00:00', '2006-1-10 00:00:00',
'asthma');

INSERT into Condition_occurrence values('3017', '1004', '2002','2000-1-1 00:00:00', '2006-1-10 00:00:00',
'asthma');

INSERT into Condition_occurrence values('3018', '1005', '2003','2000-1-1 00:00:00', '2006-1-10 00:00:00',
'asthma');
INSERT into Condition_occurrence values('3019', '1006', '2004','2000-1-1 00:00:00', '2006-1-10 00:00:00',
'asthma');

INSERT into Condition_occurrence values('3020', '1007', '2005','2000-1-1 00:00:00', '2006-1-10 00:00:00',
'asthma');
```

4. Create Drug_exposure table

```
Create table Drug_exposure(
Drug_exposure_id int NOT NULL,
Person_id int NULL,
Visit_occurrence_id int NULL,
Drug_exposure_start_date datetime NULL,
Drug_exposure_end_date datetime NULL,
Drug_name varchar(225) NULL,
CONSTRAINT PK_Drug PRIMARY KEY (Drug_exposure_id),
CONSTRAINT FK_Drug FOREIGN KEY (Person_id)
        REFERENCES Person(Person_id),
CONSTRAINT FK_Drug_Visit FOREIGN KEY (Visit_occurrence_id)
REFERENCES Visit_occurrence(Visit_occurrence_id) ON UPDATE
CASCADE
);
```

Insert synthetic data (~5 rows or more)

```
INSERT into Drug_exposure VALUES( '4001', '1001','2001','2000-1-1 00:00:00', '2000-1-10 00:00:00','ab');
INSERT into Drug_exposure VALUES( '4002', '1001','2002','2000-1-2 00:00:00', '2000-1-12 00:00:00','bc');
INSERT into Drug_exposure VALUES( '4003', '1001','2003','2000-1-3 00:00:00', '2000-1-13 00:00:00','cd');
INSERT into Drug_exposure VALUES( '4004', '1002','2004','2000-1-4 00:00:00', '2000-1-14 00:00:00','ad');
INSERT into Drug_exposure VALUES( '4005', '1002','2005','2000-1-5 00:00:00', '2000-1-15 00:00:00','be');
```

5. Create Procedure_occurrence table

```
Create table Procedure_occurrence(
Procedure_occurrence_id int NOT NULL,
Person_id int NULL,
Visit_occurrence_id int NULL,
Procedure_date datetime NULL,
Procedure_name varchar(225) NULL,
CONSTRAINT PK_Pro PRIMARY KEY (Procedure_occurrence_id),
CONSTRAINT FK_Pro FOREIGN KEY (Person_id)
```

```
        REFERENCES Person(Person_id),
CONSTRAINT FK_Pro_Visit FOREIGN KEY(Visit_occurrence_id)
REFERENCES Visit_occurrence(Visit_occurrence_id) ON UPDATE
CASCADE


);
```

Insert synthetic data (~5 rows or more)

```
INSERT into Procedure_occurrence values( '5001', '1001','2001','2000-1-1 00:00:00','abc');
INSERT into Procedure_occurrence values( '5002', '1001','2002','2000-1-2 00:00:00','bcc');
INSERT into Procedure_occurrence values( '5003', '1001','2003','2000-1-3 00:00:00','cdd');
INSERT into Procedure_occurrence values( '5004', '1002','2004','2000-1-4 00:00:00','add');
INSERT into Procedure_occurrence values( '5005', '1002','2005','2000-1-5 00:00:00','bee');
```

6. Create Sub-topic table

```
Create table Air_quality(
Record_id int NOT NULL,
FIPScode int NULL,
StateName varchar(225) NULL,
CountyName varchar(225) NULL,
ReportYear int NULL,
MeasureType int NULL,
MeasureValue int NULL,
CONSTRAINT PK_Air PRIMARY KEY (Record_id)
);
```

The data is imported from public data sources.

Appendix 2 Python Code about Transforming Data from SQL to JSON

```
import pymssql
import pyodbc
import json
import os

# create a json file
JSON_FILE = os.path.expanduser("C:/Users/Xu Chen/\Desktop\homework\SQLtoJSON.json")
# conncect to sql server
connection = pyodbc.connect('Driver={SQL Server Native Client
11.0};''Server=PC204978\SQLEXPRESS;''Database=master;''Trusted_Connection=yes;')
cur = connection.cursor()
# input person_id
cur.execute("select Person_id from person;")
resultlist_patient = cur.fetchall()
# input demographic information
cur.execute(
"""SELECT Person_id,Gender,Race,Year_of_birth,FIPScode
FROM Person p

""")
resultlist_demo = cur.fetchall()

# input procedure information
cur.execute(
"""select person_id,visit_occurrence_id, procedure_date,procedure_name
    from Procedure_occurrence""")
resultlist_ce = cur.fetchall()

# input visit information
cur.execute(
"""select person_id,visit_occurrence_id,visit_start_date,visit_end_date,visit_type
from visit_occurrence""")
resultlist_ap = cur.fetchall()

# input condition information
cur.execute(
"""select person_id,visit_occurrence_id,condition_start_date, condition_name
    from condition_occurrence""")
```

```
resultlist_cd = cur.fetchall()

# input drug information
cur.execute(
"""select person_id,visit_occurrence_id,drug_exposure_start_date,drug_exposure_end_date,drug_name
    from drug_exposure """)
resultlist_drug = cur.fetchall()

class Patient:
    def __init__(self, parameters): #a way to instantiate the class; self is the instance of the
class
        self.acct = parameters[0]
class Demographics:
    def __init__(self, parameters):
        self.acct = parameters[0]
        self.sex = parameters[1]
        self.race = parameters[2]
        self.age = parameters[3]
        self.caresite = parameters[4]
class Appointment:
    def __init__(self, parameters):
        self.acct = parameters[0]
        self.visitid = parameters[1]
        self.date = str(parameters[2])
        self.enddate = str(parameters[3])
        self.type = parameters[4]

class Clinical_event:
    def __init__(self, parameters):
        self.acct = parameters[0]
        self.visitid = parameters[1]
        self.date = str(parameters[2])
        self.event = parameters[3]
class Diagnosis:
    def __init__(self, parameters):
        self.acct = parameters[0]
        self.visitid = parameters[1]
        self.date = str(parameters[2])
        self.diag = parameters[3]
class Drug:
    def __init__(self, parameters):
```

```
        self.acct = parameters[0]
        self.visitid = parameters[1]
        self.startdate = str(parameters[2])
        self.enddate = str(parameters[3])
        self.drug = parameters[4]

# use person_id as index to create json data structure
CE={}
for res in resultlist_patient:
    p = Patient(res)
    CE[p.acct] = dict()
    CE[p.acct]['appt'] = dict()
    CE[p.acct]['sex'] = 'NA'
    CE[p.acct]['race'] = 'NA'
    CE[p.acct]['YOB'] = -1
    CE[p.acct]['visitIDlist'] = []

# input demographic information when person_id is matched
for res in resultlist_demo:
    demo = Demographics(res)
    try:
        CE[demo.acct]['sex'] = demo.sex
        CE[demo.acct]['YOB'] = int(demo.age)
        CE[demo.acct]['race'] = demo.race
    except:
        pass
# input visit information when person_id is matched
for res in resultlist_ap:
    ap = Appointment(res)
    if ap.acct in CE:
        CE[ap.acct]['appt'][ap.date] = dict()
        CE[ap.acct]['appt'][ap.date]['ID'] = ap.visitid
        CE[ap.acct]['appt'][ap.date]['actualdate'] = ap.date
        CE[ap.acct]['appt'][ap.date]['enddate'] = ap.enddate
        CE[ap.acct]['appt'][ap.date]['type'] = ap.type
        CE[ap.acct]['appt'][ap.date]['diag'] = list()
        CE[ap.acct]['appt'][ap.date]['drug'] = list()
        CE[ap.acct]['appt'][ap.date]['proc'] = list()

# input procedure information when person_id is matched
for res in resultlist_ce:
```

```
    ce = Clinical_event(res)
    if ce.acct in CE:
        for date in CE[ce.acct]['appt']:
            if ce.visitid == CE[ce.acct]['appt'][date]['ID']:
                CE[ce.acct]['appt'][date]['proc'].append(ce.event)
                CE[ce.acct]['appt'][date]['proc'] = sorted(CE[ce.acct]['appt'][date]['proc'])

# input condition information when person_id is matched
for res in resultlist_cd:
    cd = Diagnosis(res)
    if cd.acct in CE:
        for date in CE[cd.acct]['appt']:
            if cd.visitid == CE[cd.acct]['appt'][date]['ID']:
                CE[cd.acct]['appt'][date]['diag'].append(cd.diag)
                CE[cd.acct]['appt'][date]['diag'] = sorted(CE[cd.acct]['appt'][date]['diag'])


# input drug information when person_id is matched
for res in resultlist_drug:
    cdrg = Drug(res)
    if cdrg.acct in CE:
        for date in CE[cdrg.acct]['appt']:
            if cdrg.visitid == CE[cdrg.acct]['appt'][date]['ID']:
                d = dict()
                d['name'] = cdrg.drug
                d['startdate'] = cdrg.startdate
                CE[cdrg.acct]['appt'][date]['drug'].append(d)

# get the json file
with open(JSON_FILE, 'w') as outfile:
    json.dump(CE, outfile, indent=2, sort_keys=True, separators=(',', ': '))
```