

***DETERIORATING QUALITY OF AIR QUALITY***



**Table of Contents**

Introduction1

File v/s DBMS4

Relational Model5

ER Model6

**REPORT**

**Introduction**

We often read in newspapers about the deteriorating air quality of the metro cities. This has become the major concern recently and several steps are being taken in accordance with its improvement. The Environmentalist has declared an alarming situation in national capital of Delhi.

What is DBMS?

Database management system is a file handling software used for storing data and retrieving it whenever required. This system has helped us immensely in collecting and storing data in an organized format. These are easily accessible.

**File System v/s DBMS**

## DATA REDUNDANCY

1. Data redundancy occurs when the same piece of data exists at several points or places at a time. Same data is stored at multiple locations.
2. The air quality data is collected at the various stations. They store their own data and also share it with the centres at the state levels. This data is again shared by the state at the national level.
3. If the need arises that some other department has to access the same data, the same data has to be shared all over again.
4. Sensors at the stations keep on updating the data almost every minute, which leads to generation of huge amounts of data.
5. If any private institutions or individuals required data for research purposes then, the data has to be sent to them individually.

## DATA INCONSISTENCY

1. Data at different places is supposed to be the same but due to negligence or error the data differs from each other.
2. If the data of the sensors are being stored in the centralised system, then this is easier to manage.
3. For example, today’s data set has been sent to the state level but while forwarding it to the national station, the state level in charge sends yesterdays’ data. Now because of this the data at the national level for the respective state would be duplicated for two consecutive days.
4. For each sensor a different table has to be created when implementing a file system.
5. If some failure occurs while sending the files from the station to the state level then the files have to be shared all over again.

## EASY ACCESS TO DATA

1. The sensors generate a lot of data as it is working every minute.
2. The data will be updated every minute which would be difficult to be analysed manually.
3. Hence any GUI System is created for visualising and analysing data which can work with a centralised database and the memory is easily accessible.
4. The data can be shared through API directly with the institutions or individuals for research purposes.
5. Various insights will be available to us because of easy access which would help in stopping the deteriorating conditions of the environment.

## DATA ISOLATION

1. The latest updated data will not be available.
2. In case a programmer writes a code, then he’ll not receive the latest data rather he’ll have to work with stale data.
3. If there is a centralised data, then a single code can be created to analyse data from all the different levels and stations.
4. Programmers at different levels will have to write their own individual codes.
5. The files at the station level may have different formats, then there will be a problem in processing the data.

## DATA INTEGRITY

1. The government can be a main cause of data being hampered. They can alter the data to demonstrate a clean image in front of the public.
2. If DBMS is used then, this can surely be avoided and the data integrity can be maintained.
3. As we know that the sensors are the one going to write the data and hence all users should only be granted read access.
4. In the file system any one can modify the data for whatever reasons.
5. The concept of foreign keys cannot be implemented in a file system and hence the data from various stations if necessary, cannot be grouped together based on date, time, etc. Example : If every hour we want to calculate city and state averages for the concentration levels the data from all stations has to be grouped and then it has to be calculated.

## ATOMICITY OF UPDATES

1. If the sensors have got into some malfunctioning, then Null data will be updated in the file system continuously which cannot be altered.
2. This would create a serious problem in the analysis system. If suppose the ozone sensor has stopped working, then the file system will receive Null values till the time it is repaired. This would alter the data analysis.
3. Hence if a database is used then a condition can be put up such that if a sensor returns NULL values then those should be discarded and some other values should be substituted instead using some statistical analysis, based on time and location a time series analysis can be performed.
4. If any transaction has just been completed and power failure occurs, still the transaction has been saved. But if a power failure occurs while writing the data in a file then it has to be saved first to save the transactions permanently.

## SECURITY

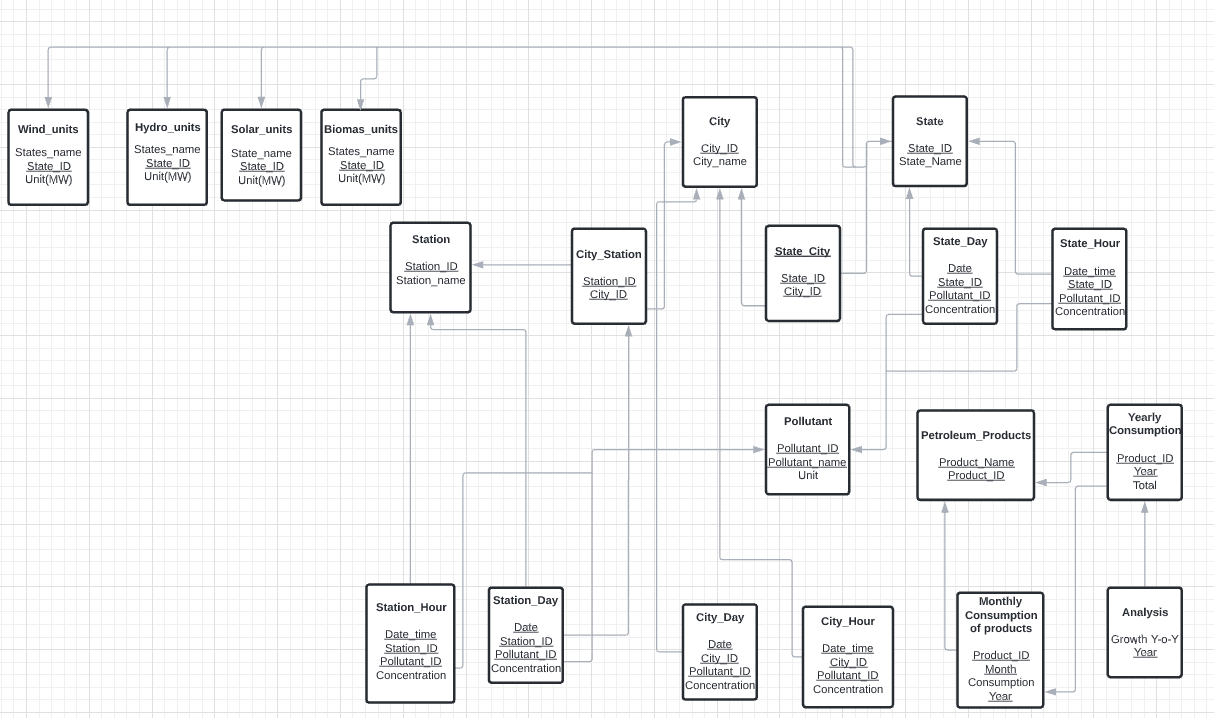
1. In the case of a file system, the data will be stored in separate computers in various stations. That computer will be having several users. Hence, data can be easily hampered or altered.
2. Also, the computers can be hacked easily. Hundreds of computers cannot be provided with high security.
3. In DBMS the central system will be providing high security. All the data will be controlled using a single security system and the legitimacy of data can be thus maintained.
4. No user needs to have an update access on the data.
5. In a database system all stations can view only the data shared by their respective station and same with the state level. For accessing the whole database they too would have to use an API.

## CONCURRENT ACCESS

1. Several sensors can write data at the same time in a DBMS system. This is not possible in the file system. In that case we’ll have to compile system files separately.
2. In a DBMS system all the sensors can write the data at the same place and at the same time. The data from all the stations, states and nations can be combined, stored and accessed from this system itself. Whereas in the case of a file system the data has to be individually combined at different levels and then be shared.
3. The data stored in the file system will be available to only a few people. If some other person wants access to it, then he’ll have to undergo long procedures in order to receive the data.
4. If we create an API of the dataset, then the users throughout the nation can easily access it and get the latest updated data.

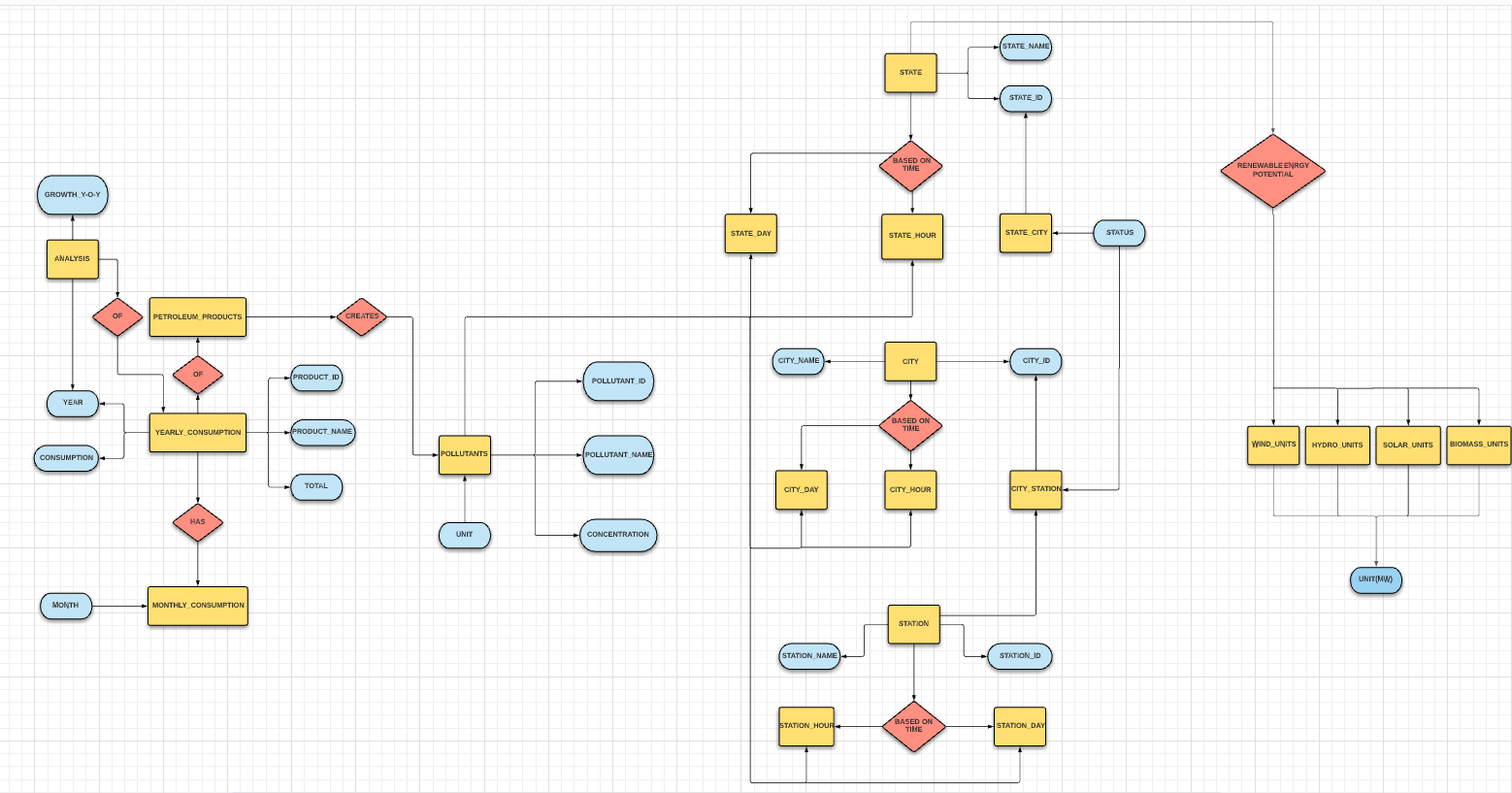
**RELATIONAL MODEL**

Relational Model is the data representation that demonstrates the relation in tables. The rows of the table are connected with some real-world relation.



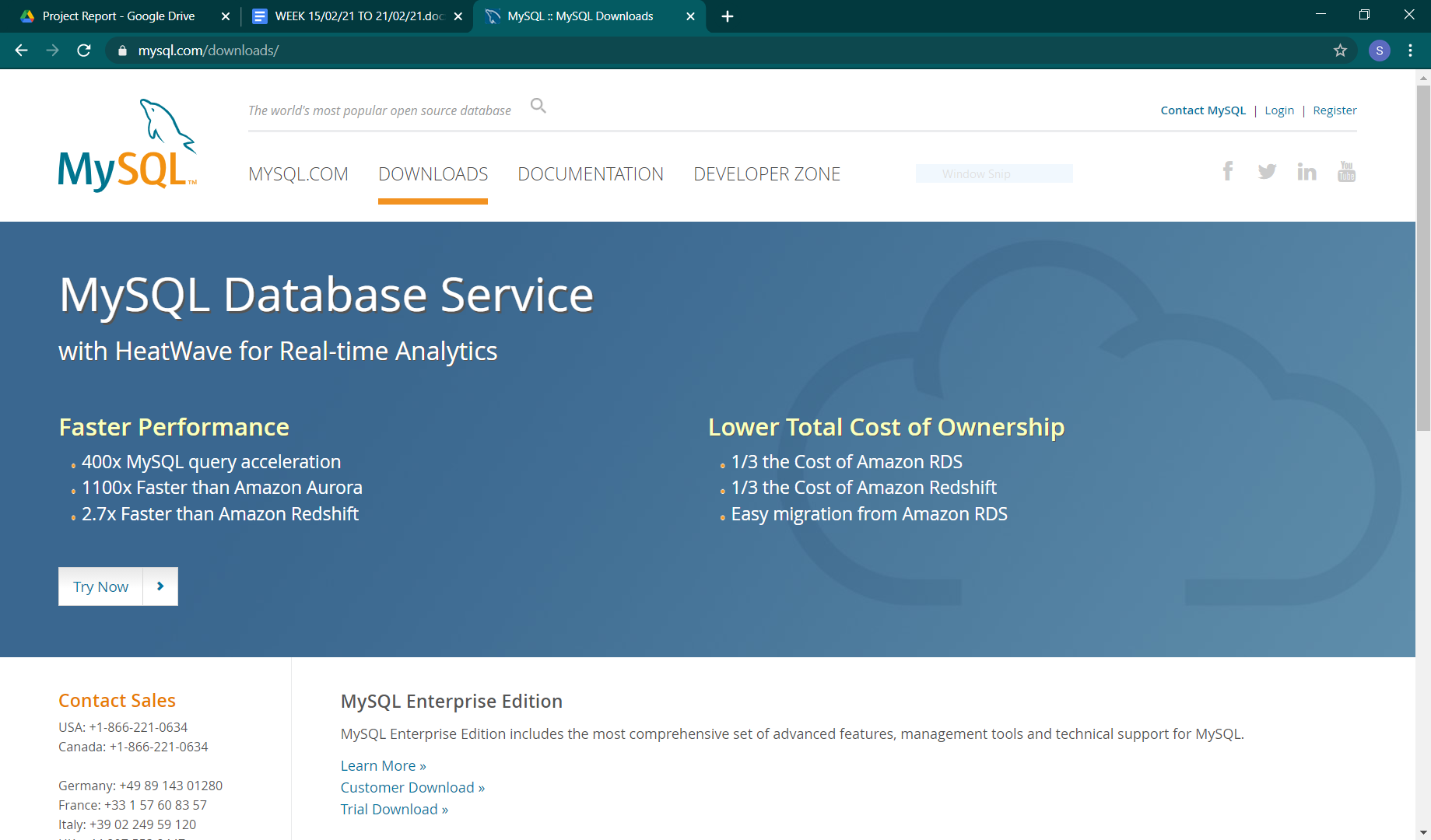
**ER MODEL**

The entity relationship diagram is basically considered to be the blueprint of the database which is finally converted to a database. It defines the relationship between elements and is a high-level data model.



|  |  |  |  |
| --- | --- | --- | --- |
| **Table name** | **Super key** | **Candidate key** | **Primary key** |
| Station | Station\_Id, Station\_name, [Station\_Id, station name] | Station\_ID,State\_name | Station\_ID |
| City\_Station | [Station\_ID, City\_ID],  [Station\_ID, City\_ID] | [Station\_ID, City\_ID] | [Station\_ID, City\_ID] |
| City | City\_ID, City\_name, [City\_ID, City\_name] | City\_ID, City\_name | City\_ID |
| State | State\_ID, State\_name | State\_ID, State\_name | State\_ID |
| State\_City | State\_ID, City\_ID, [State\_ID, City\_ID] | [State\_ID, City\_ID] | [State\_ID, City\_ID] |
| State\_Day | [Date, Pollutant\_ID, State\_ID] | [Date, Pollutant\_ID, State\_ID ] | [Date, State\_ID,Pollutant\_ID] |
| State\_ Hour | [Date\_Time, State\_ID, Pollutant\_ID] | [Date\_Time, State\_ID, Pollutant\_ID] | [Date\_Time, State\_ID, Pollutant\_ID] |
| Pollutant | Pollutant\_ID, Pollutant\_name, [Pollutant\_ID, Pollutant\_name],  [Pollutant\_ID,Pollutant\_name,  unit],[Pollutant\_ID,Unit],  [Pollutant\_name,Unit] | Pollutant\_ID, Pollutant\_name | Pollutant\_ID |
| Station Hour | [Date\_time, Pollutant\_ID, Station\_ID]  [Date\_time,Pollutant\_ID,Station\_ID,concentration] | [Date\_time,Station\_ID, Pollutant\_ID] | [Date\_time,Station\_ID, Pollutant\_ID] |
| Station Day | [Date,Station\_ID,Pollutant\_ID],  [Date,Station\_ID,Pollutant\_ID,concentration] | [Date,Station\_ID,Pollutant\_ID] | [Date,Station\_ID,Pollutant\_ID] |
| City Hour | [Date\_time, Pollutant\_ID, City\_ID]  [Date\_time,Pollutant\_ID,City\_ID,concentration] | [Date\_time,City\_ID, Pollutant\_ID] | [Date\_time,City\_ID, Pollutant\_ID] |
| City Day | [Date,City\_ID,Pollutant\_ID],  [Date,City\_ID,Pollutant\_ID,concentration] | [Date,City\_ID, Pollutant\_ID] | [Date,City\_ID, Pollutant\_ID] |
| Petroleum Products | Product\_Name, Product\_ID  [Product\_ID,Product\_Name] | Product\_Name, Product\_ID | Product\_ID |
| Yearly Consumption | [Product\_ID, Year] | [Product\_ID, Year] | [Product\_ID,Year] |
| Monthly Consumption | Product\_ID, Month, Year, [Product\_ID, Year], [Month, Year, Product\_ID] | Product\_ID, Year, Month | Product\_ID |
| Analysis | Year,[Year,Growth Y-o-Y] | Year | Year |
| Wind Units | State\_ID, State\_name, [State\_ID, State\_name] | State\_ID, State\_name | State\_ID |
| Biomass Units | State\_ID, State\_name, [State\_ID, State\_name] | State\_ID, State\_name | State\_ID |
| Hydro Units | State\_ID, State\_name, [State\_ID, State\_name] | State\_ID, State\_name | State\_ID |
| Solar Units | State\_ID, State\_name, [State\_ID, State\_name] | State\_ID, State\_name | State\_ID |

**INSTALLATION OF SOFTWARE**



We have downloaded and installed MySQL software for performing our query operations in DBMS.

Following are the steps to be followed:

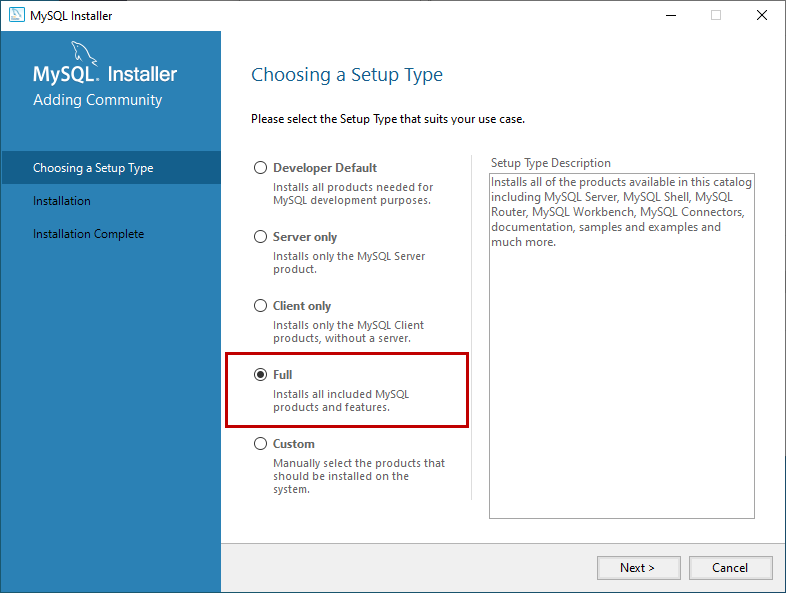
Step1

Download the Mysql software from MySql.com

Step2

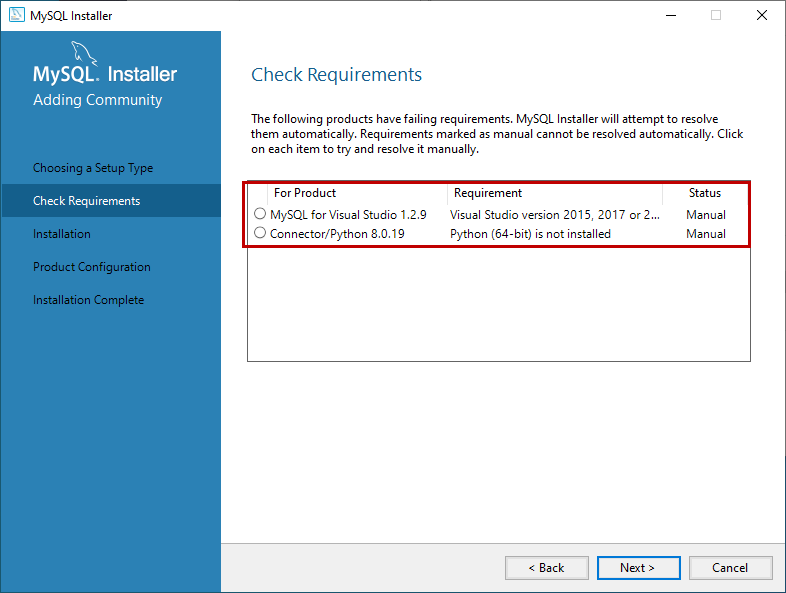
Extract the files from the zip and run the setup.

Step3



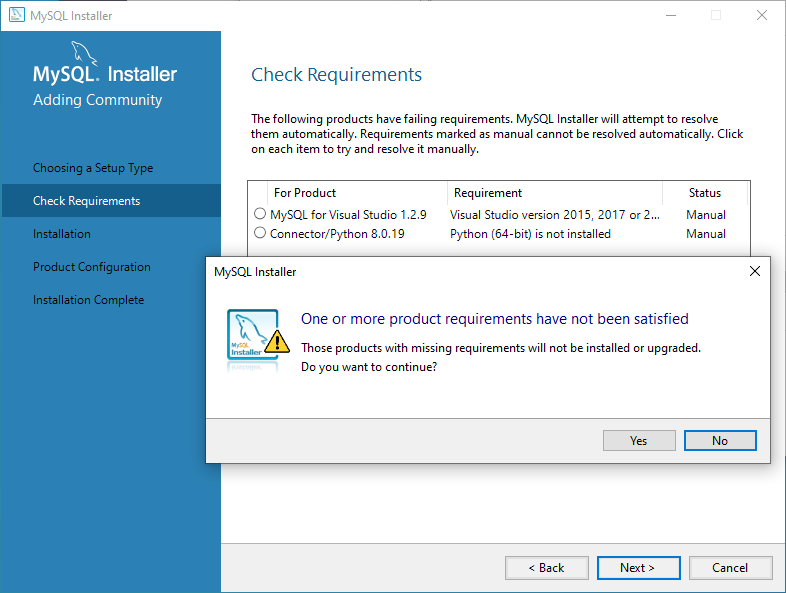
Step4

Before installation the software would check for the libraries required and whether the settings are appropriate or not.



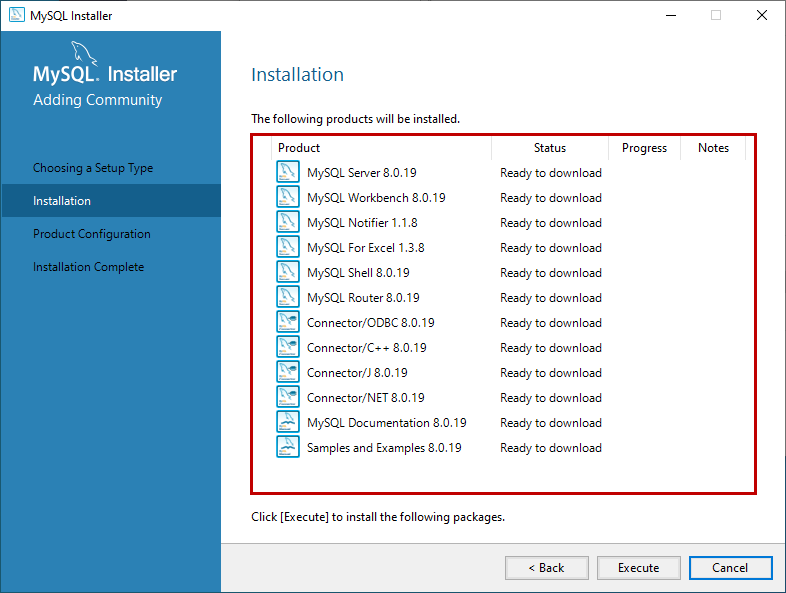
Step5

A pop up window appears giving us a warning regarding the presence of VSC or Python. Click on Yes..



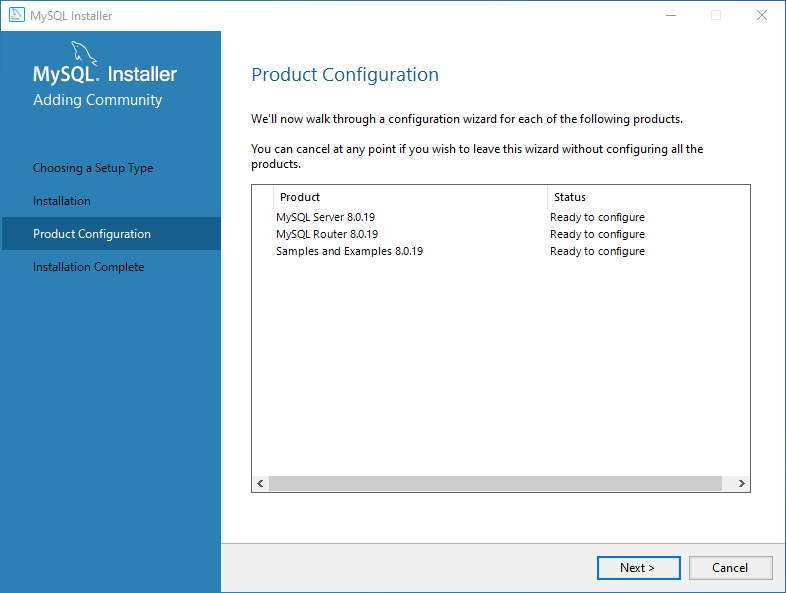
Step6

Then Click on Execute to install all the packages.



Step7

After the installation of packages is completed. A dialog box appears. Then Click on Next to continue.



Step8

After this step your Sql would be ready to resolve the queries. Just click on Finish, to complete the installation process.

# **Assignment 4**

## 1) Selection

1) To find the state which have more solar potential than the national average.

select \*

from dbms.solar

where units >(select avg(units) from dbms.solar);

2) Finding states having solar potential greater than 50000 units.

select \* from dbms.solar

where dbms.solar.units>50000;

3) Finding states with sum of solar and biomass potential greater than 40000 units and sorting by solar and biomass potential.

select dbms.solar.state\_name,dbms.solar.units as solar\_units,dbms.biomaxx.units as biomass\_units,dbms.solar.units+dbms.biomaxx.units as Total

from dbms.biomass right join dbms.solar on dbms.solar.state\_id=dbms.biomass.state\_id

where dbms.solar.units+dbms.biomass.units>40000

order by Total desc;

4) Finding those states which occur in the daily pollution data.

select state.state\_name

from state

where exists(select \* from state\_day where state\_day.state\_id=state.state\_id);

5) Finding those stations in cities which exist both in the state\_city table and the city\_station table.

select station.station\_name

from station

where exists (select \*

from city\_station

where exists(select \* from state\_city where state\_city.city\_id=city\_station.city\_id));

## 2) Projection

1) Solar units potential by each state.

select state\_name,units from dbms.solar;

2) Pollution data of different cities.

select city\_id,date,pollutant,concentration from dbms.city\_day;

3) Rounding and finding the square root of the daily state wise concentration of pollutants.

select state\_id,pollutant\_id,round(concentration),round(sqrt(concentration),2)

from state\_day;

4) Finding those states which occur in the daily pollution data.

select state.state\_name

from state

where exists(select \* from state\_day where state\_day.state\_id=state.state\_id);

5) Finding those stations in cities which exist both in the state\_city table and the city\_station table.

select station.station\_name

from station

where exists (select \*

from city\_station

where exists(select \* from state\_city where state\_city.city\_id=city\_station.city\_id));

## 

## 

## 3) Cartesian Product

1) Cartesian product between the solar and the hydro table.

select \*

from dbms.solar,dbms.hydro;

2) Cartesian Product between the Station and City pollutant

Select \*

From City;

Select CityId.Station\_day, CityId.City\_day

Where concentration> AVG(concentration);

3) To perform cartesian product of wind table and hydro table

SELECT \* from wind\_dbms

CROSS JOIN hydro;

4)To find state that has solar and wind power potential is more than 10000

SELECT State, Solar energy, Wind energy

FROM solar, wind

WHERE Solar energy>10000 AND Wind energy>10000;

5) To view city day data and state day data together

SELECT \*

FROM City\_day.dbms, State\_day.dbms

## 

## 4) Union

1) Finding states that are in the top 5 of either solar, wind, hydro and biomass potential.

(select state\_name,'solar' from solar order by units desc limit 5)

union

(select state\_name,'wind' from wind order by units desc limit 5)

union

(select state\_name,'hydro' from hydro order by units desc limit 5)

union

(select state\_name,'biomass' from biomaxx order by units desc limit 5);

2) Union of the hydro and solar table and a column to find which row came from which table.

select \*,'hydro' from dbms.hydro

union

select \*,'solar' from dbms.solar;

3) To demonstrate a state with potential solar energy and the concentration of producing pollutant

SELECT state\_id, pollutant, concentration

FROM State\_day

Left Join Solar

ON State\_Day.StateId==Solar.StateId

UNION

SELECT state\_id, pollutant, concentration

FROM State\_day

Left Join Solar

ON State\_Day.StateId==Solar.StateId;

4)To demonstrate all renewable energy together

Select \*

From hydro

Where hydro energy>AVG(hydro energy)

UNION

Select \*

From solar

Where solar energy>AVG(solar energy)

UNION

Select \*

From biomass

Where biomass energy>AVG(biomass energy)

## 5) Set Difference

1) To find states which don’t produce both hydro-power and biomass-power but only biomass power.

SELECT states

FROM hydro

LEFT JOIN

biomass USING (states)

WHERE

Biomass.states IS NULL;

2) To find filtered data

select \*

from dbms1.hydro

minus

select \*

from dbms1.solar;

3) To find the state-id of states not producing power from solar energy.

select stateid from state\_city

left join

solar using(stateid) where solar.stateid is null;

4) To find the city-id of cities which don’t produce hydro-energy.

select \* from state\_city

where

stateid=any(select stateid from state\_city

left join

hydro using(stateid)

where

hydro.stateid is null);

## 6) Natural Join

1) Natural join of the solar and hydro table

select \*

from dbms.hydro natural join dbms.solar;

## 7) Composition of 2 from above

List of states producing above average power in both solar and hydro sectors.

select states from hydro

where

hydropower>(select avg(hydropower) as power1 from hydro)

union

select state from solar

where

solarpower>(select avg(solarpower) as power2 from solar);

## 8) Composition of 3 from above

1) Taking Cartesian product of solar and wind table and selection those rows which have solar.state\_id greater than 10.

select solar.state\_name

from dbms.solar, dbms.wind

where solar.state\_id>10;

# **Assignment 6**

CREATE TABLE `Wind`(

States \_name char(30) NOT NULL,

State\_ID int(20) NOT NULL,

Units int(20) NOT NULL,

PRIMARY KEY(State\_ID)

);

CREATE TABLE `Hydro`(

States \_name char(30) NOT NULL,

State\_ID int(20) NOT NULL,

Units int(20) NOT NULL,

PRIMARY KEY(State\_ID)

);

CREATE TABLE ‘Solar’(

States \_name char(30) NOT NULL,

State\_ID int(20) NOT NULL,

Units int(20) NOT NULL,

PRIMARY KEY(State\_ID)

);

CREATE TABLE ‘Biomass’(

States \_name char(30) NOT NULL,

State\_ID int(20) NOT NULL,

Units int(20) NOT NULL,

PRIMARY KEY(State\_ID)

);

CREATE TABLE `City\_station`(

Station\_ID int(20) NOT NULL,

City\_ID int(20) NOT NULL,

);

CREATE TABLE `Station`(

Station\_ID int(20) NOT NULL,

Station\_name char(20) NOT NULL,

PRIMARY KEY(Station\_ID)

);

CREATE TABLE `city`(

City\_name char(20) NOT NULL

City\_ID int(20) NOT NULL,

PRIMARY KEY(City\_ID)

);

CREATE TABLE `State`(

State\_ID int(20) NOT NULL,

State\_name char(20) NOT NULL,

PRIMARY KEY(State\_ID)

);

CREATE TABLE `State\_city`(

State\_ID int(20) NOT NULL,

City\_ID int(20) NOT NULL,

);

CREATE TABLE `State\_day`(

Date varchar(20) NOT NULL

State\_ID int(20) NOT NULL,

Pollutant\_ID int(20) NOT NULL,

Concentration int(20) NOT NULL,

);

CREATE TABLE `State\_hour`(

Date\_time varchar(20) NOT NULL

State\_ID int(20) NOT NULL,

Pollutant\_ID int(20) NOT NULL,

Concentration int(20) NOT NULL,

);

CREATE TABLE `Station\_hour`(

Date\_time varchar(20) NOT NULL

Station\_ID int(20) NOT NULL,

Pollutant\_ID int(20) NOT NULL,

Concentration int(20) NOT NULL,

);

CREATE TABLE Station\_day``(

Date varchar(20) NOT NULL

Station\_ID int(20) NOT NULL,

Pollutant\_ID int(20) NOT NULL,

Concentration int(20) NOT NULL,

);

CREATE TABLE `City\_day`(

Date varchar(20) NOT NULL

City\_ID int(20) NOT NULL,

Pollutant\_ID int(20) NOT NULL,

Concentration int(20) NOT NULL,

);

CREATE TABLE `City\_hour`(

Date\_time varchar(20) NOT NULL

City\_ID int(20) NOT NULL,

Pollutant\_ID int(20) NOT NULL,

Concentration int(20) NOT NULL,

);

CREATE TABLE `Pollutant`(

Pollutant\_ID int(20) NOT NULL,

Pollutant\_Name varchar(20) NOT NULL

);

CREATE TABLE `Petroleum\_products`(

Product\_name varchar(50) NOT NULL,

Product\_ID int(50) NOT NULL,

);

CREATE TABLE`Yearly\_consumption`(

Product\_ID int(50) NOT NULL,

Year varchar(30) NOT NULL,

Total int(50) NOT NULL,

);

CREATE TABLE `monthly\_consumption\_of\_products`(

Product\_ID int(50) NOT NULL,

Month varchar(30) NOT NULL,

Consumption varchar(30) NOT NULL,

Year varchar(30) NOT NULL

);

CREATE TABLE `analysis`(

Growth varchar(50) NOT NULL,

Year varchar(30) NOT NULL

);

# **Assignment 7**

## 1) Aggregate functions

1) Find the state with maximum potential of solar energy.

select \*

from dbms.solar

where units in (select max(units) from dbms.solar);

2) To find the state which have more solar potential than the national average.

select \*

from dbms.solar

where units >(select avg(units) from dbms.solar);

3) Find the total number of stations that are measuring the data.

select count(\*) as total\_stations from city\_station;

4) Finding average concentration of PM2.5 in each month for all states.

select state\_id,substring(state\_day.date,7,4) as year,substring(state\_day.date,4,2) as month,avg(concentration),count(concentration)

from state\_day

where state\_day.pollutant\_id='PM2.5'

group by state\_day.state\_id,2,3;

5) Finding average pollution of different pollution month wise for all the cities.

select city\_id,city\_day.pollutant,substring(city\_day.date,7,4) as year,substring(city\_day.date,4,2) as month,avg(concentration),count(concentration)

from city\_day

group by city\_day.pollutant,city\_day.city\_id,2,3;

## 2) Built in numeric functions

1) Rounding and finding the square root of the daily state wise concentration of pollutants.

select state\_id,pollutant\_id,round(concentration),round(sqrt(concentration),2)

from state\_day;

## 3) Built in string functions

1) Finding average concentration of PM2.5 in each month for all states.

select state\_id,substring(state\_day.date,7,4) as year,substring(state\_day.date,4,2) as month,avg(concentration),count(concentration)

from state\_day

where state\_day.pollutant\_id='PM2.5'

group by state\_day.state\_id,2,3;

2) Finding average pollution of different pollution month wise for all the cities.

select city\_id,city\_day.pollutant,substring(city\_day.date,7,4) as year,substring(city\_day.date,4,2) as month,avg(concentration),count(concentration)

from city\_day

group by city\_day.pollutant,city\_day.city\_id,2,3;

3) Converting city name to upper case.

select ucase(City\_Name)

from dbms.city;

4) Find length of city names and use div function to divide them by 2.

select length(City\_Name) div 2,City\_Name

from dbms.city;

# **Assignment 8**

## 1) Set operations

1) Finding states with sum of solar and biomass potential greater than 40000 units and sorting by solar and biomass potential.

select dbms.solar.state\_name,dbms.solar.units as solar\_units,dbms.biomaxx.units as biomass\_units,dbms.solar.units+dbms.biomaxx.units as Total

from dbms.biomaxx right join dbms.solar on dbms.solar.state\_id=dbms.biomaxx.state\_id

where dbms.solar.units+dbms.biomaxx.units>40000

order by Total desc;

2) Finding number of stations in each cities.

select city.City\_Name,count(\*) as num\_station\_in\_city

from city\_station left join city

on city.City\_ID=city\_station.city\_id

group by city.City\_Name

order by 2 desc;

3) Number of stations in each state.

select state.state\_name,count(\*)

from city left join city\_station on city.City\_ID=city\_station.city\_id join state\_city on state\_city.city\_id=city.City\_ID inner join state on state\_city.state\_id=state.state\_id

group by state\_city.state\_id

order by 2 desc;

4) Number of cities in each states which has a station to measure air quality.

select state.state\_name,count(\*) as num\_city\_in\_each\_state,state.state\_id

from state\_city right join state

on state\_city.state\_id=state.state\_id

group by state.state\_id

order by 2 desc;

5) Find the sum of the solar,hydro,wind and biomass potential of all states.

select solar.state\_name,solar.units as solar\_units,hydro.units as hydro\_units,wind.units as wind\_units,biomaxx.units as biomass\_units,

biomaxx.units+solar.units+wind.units+hydro.units as total

from biomaxx join solar on biomaxx.state\_id=solar.state\_id join hydro on solar.state\_id=hydro.state\_id join wind on wind.state\_id=solar.state\_id

order by 6 desc;

6) Finding those states which are present both in the states list as well in the state\_city table.

select state.state\_name

from state inner join state\_city using (state\_id);

## 2) Sub queries

1) Find the state with maximum potential of solar energy.

select \*

from dbms.solar

where units in (select max(units) from dbms.solar);

2) To find the state which have more solar potential than the national average.

select \*

from dbms.solar

where units >(select avg(units) from dbms.solar);

3) Finding the sum of the solar, hydro, wind and biomass potential of all states which have wind potential greater than the national average.

select solar.state\_name,solar.units as solar\_units,hydro.units as hydro\_units,wind.units as wind\_units,biomaxx.units as biomass\_units,

biomaxx.units+solar.units+wind.units+hydro.units as total

from biomaxx join solar on biomaxx.state\_id=solar.state\_id join hydro on solar.state\_id=hydro.state\_id join wind on wind.state\_id=solar.state\_id

where wind.units > (select avg(wind.units) from wind)

order by 4 desc;

4) Deleting entries of solar power with potential less than average value

Select solar.state, solar.units as State, Units

From Solar

Delete Solar.state, Solar.units (Where Solar.units<Avg(Solar.units))

## 3) Correlated sub queries

1) Selecting only those states which exist in wind and biomaxx table. (Intersect)

select \*

from dbms.wind

where wind.units >(select biomaxx.units

from biomaxx

where biomaxx.state\_id=wind.state\_id);

2)

select \*

from state\_day as st left join state on state.state\_id=st.state\_id

where state.state\_id= (select state\_id

from state

where st.state\_id=state.state\_id)

3) Finding those cities which occur in the daily data.

select city.City\_Name

from city

where exists(select \*

from city\_day

where city.City\_ID=city\_day.city\_id);

4) Finding those stations in cities which exist both in the state\_city table and the city\_station table.

select station.station\_name

from station

where exists (select \*

from city\_station

where exists(select \* from state\_city where state\_city.city\_id=city\_station.city\_id));

5) Finding those states which are in the solar and the state table.

select state.state\_name

from state

where exists(select \*

from solar

where solar.state\_name=state.state\_name);

# **Assignment 9**

## 1) Group by

1) Finding number of stations in each cities.

select city.City\_Name,count(\*) as num\_station\_in\_city

from city\_station left join city

on city.City\_ID=city\_station.city\_id

group by city.City\_Name

order by 2 desc;

2) Number of cities in each states which has a station to measure air quality.

select state.state\_name,count(\*) as num\_city\_in\_each\_state,state.state\_id

from state\_city right join state

on state\_city.state\_id=state.state\_id

group by state.state\_id

order by 2 desc;

3) Number of stations in each state.

select state.state\_name,count(\*)

from city left join city\_station on city.City\_ID=city\_station.city\_id join state\_city on state\_city.city\_id=city.City\_ID inner join state on state\_city.state\_id=state.state\_id

group by state\_city.state\_id

order by 2 desc;

4) Find the sum of the solar,hydro,wind and biomass potential of all states.

select solar.state\_name,solar.units as solar\_units,hydro.units as hydro\_units,wind.units as wind\_units,biomaxx.units as biomass\_units,

biomaxx.units+solar.units+wind.units+hydro.units as total

from biomaxx join solar on biomaxx.state\_id=solar.state\_id join hydro on solar.state\_id=hydro.state\_id join wind on wind.state\_id=solar.state\_id

order by 6 desc;

5) Finding average concentration of PM2.5 in each month for all states.

select state\_id,substring(state\_day.date,7,4) as year,substring(state\_day.date,4,2) as month,avg(concentration),count(concentration)

from state\_day

where state\_day.pollutant\_id='PM2.5'

group by state\_day.state\_id,2,3;

## 2) Order by

1) Ranking the states by their solar potential.

select state\_name,units,rank() over(order by units desc) as solar\_rank

from dbms.solar ;

2) Finding states that are in the top 5 of either solar, wind, hydro and biomass potential.

(select state\_name,'solar' from solar order by units desc limit 5)

union

(select state\_name,'wind' from wind order by units desc limit 5)

union

(select state\_name,'hydro' from hydro order by units desc limit 5)

union

(select state\_name,'biomass' from biomaxx order by units desc limit 5);

3) Finding states with sum of solar and biomass potential greater than 40000 units and sorting by solar and biomass potential.

select dbms.solar.state\_name,dbms.solar.units as solar\_units,dbms.biomaxx.units as biomass\_units,dbms.solar.units+dbms.biomaxx.units as Total

from dbms.biomaxx right join dbms.solar on dbms.solar.state\_id=dbms.biomaxx.state\_id

where dbms.solar.units+dbms.biomaxx.units>40000

order by Total desc;

4) Finding number of stations in each cities.

select city.City\_Name,count(\*) as num\_station\_in\_city

from city\_station left join city

on city.City\_ID=city\_station.city\_id

group by city.City\_Name

order by 2 desc;

5) Find the sum of the solar,hydro,wind and biomass potential of all states.

select solar.state\_name,solar.units as solar\_units,hydro.units as hydro\_units,wind.units as wind\_units,biomaxx.units as biomass\_units,

biomaxx.units+solar.units+wind.units+hydro.units as total

from biomaxx join solar on biomaxx.state\_id=solar.state\_id join hydro on solar.state\_id=hydro.state\_id join wind on wind.state\_id=solar.state\_id

order by 6 desc;

# **Assignment 10**

## 1) Exist

1) Finding those states which occur in the daily pollution data.

select state.state\_name

from state

where exists(select \* from state\_day where state\_day.state\_id=state.state\_id);

2) Finding those states which occur in the hourly data.

select state.state\_name

from state

where exists(select \* from state\_hour where state\_hour.state\_id=state.state\_id);

3) Finding those states which are in the solar and the state table.

select state.state\_name

from state

where exists(select \*

from solar

where solar.state\_name=state.state\_name);

4) Finding those cities which occur in the daily data.

select city.City\_Name

from city

where exists(select \*

from city\_day

where city.City\_ID=city\_day.city\_id);

5) Finding those stations in cities which exist both in the state\_city table and the city\_station table.

select station.station\_name

from station

where exists (select \*

from city\_station

where exists(select \* from state\_city where state\_city.city\_id=city\_station.city\_id));

## 2) Any

1)

select \*

from city\_day

where city\_day.city\_id= any(select city\_id

from city

where city\_day.city\_id=city.City\_ID);

2)

select city.City\_Name

from city

where city.City\_Id = any(select city\_day.city\_id

from city\_day

where city\_day.concentration>28);

3)

select state.state\_name

from state

where state.state\_id = any(select state\_day.state\_id

from state\_day

where state\_day.concentration>28);

4)

select \*

from state\_day

where state\_day.concentration=any(select state\_hour.concentration

from state\_hour

where state\_hour.concentration>50);

5)

select \*

from state\_hour

where state\_hour.concentration=all(select state\_day.concentration

from state\_day

where state\_hour.concentration>50);

## 3) All

1)

select \*

from city\_day

where city\_day.city\_id= all(select city\_id

from city

where city\_day.city\_id=city.City\_ID);

2)

select city.City\_Name

from city

where city.City\_Id = all(select city\_day.city\_id

from city\_day

where city\_day.concentration>28);

3)

select state.state\_name

from state

where state.state\_id = all(select state\_day.state\_id

from state\_day

where state\_day.concentration>28);

4)

select \*

from state\_day

where state\_day.concentration=all(select state\_hour.concentration

from state\_hour

where state\_hour.concentration>50);

5)

select \*

from state\_hour

where state\_hour.concentration=all(select state\_day.concentration

from state\_day

where state\_hour.concentration>50);