PANDIT DEENDAYAL ENERGY UNIVERSITY

Formerly Pandit Deendayal Petroleum University (PDPU)

UGC RECOGNIZED

DETERIORATING QUALITY OF AIR QUALITY

Team Members:

Nandini Bhuva……………………………19BIT085

Nidhay Pancholi…………………………19BIT087

Nidhi Zala………………………………19BIT088

Shambhavi Kumar………………………19BIT115

Vivek Devre……………………………19BIT131

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 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 keeps on updating the data almost every minute, which leads to generation of huge amount of data.
5. If any private institutions or individual required data for research purpose 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 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 if any GUI System is created for visualising and analysing data which can work with centralised database and the memory is easily accessible.
4. The data can be shared through API directly with the institutions or individuals for research purpose.
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 get 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 format, 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 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 a read access.

4. In file system any one can modify the data for whatever reasons.
5. The concept of foreign keys cannot be implemented in 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 analysis system. If suppose the ozone sensors 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 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 case of 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 single security system and the legitimacy of data can be thus maintained. 4. No user needs to have an update access on the data.

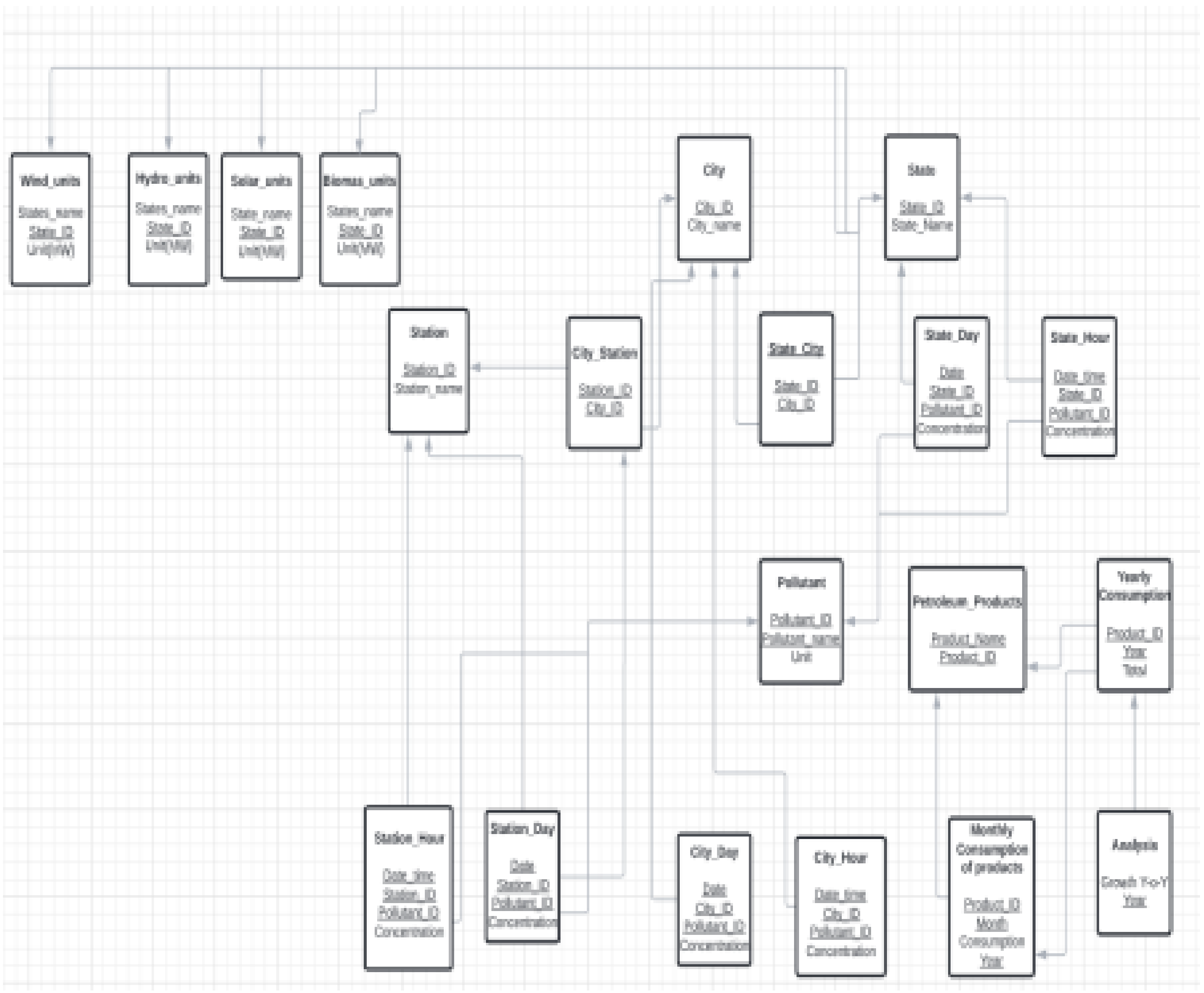
5. In a database system all stations can view only the data shared by their respective station and same with the state level. For accessing the whole database they too would have to use an API.

CONCURRENT ACCESS

1. Several sensors can write data at the same time in DBMS system. This is not possible in file system. In that case we'll have to compile system files separately.
2. In DBMS system all the sensors can write the data at 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 case of 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 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 database. It defines the relationship between elements and is a high-level data model.

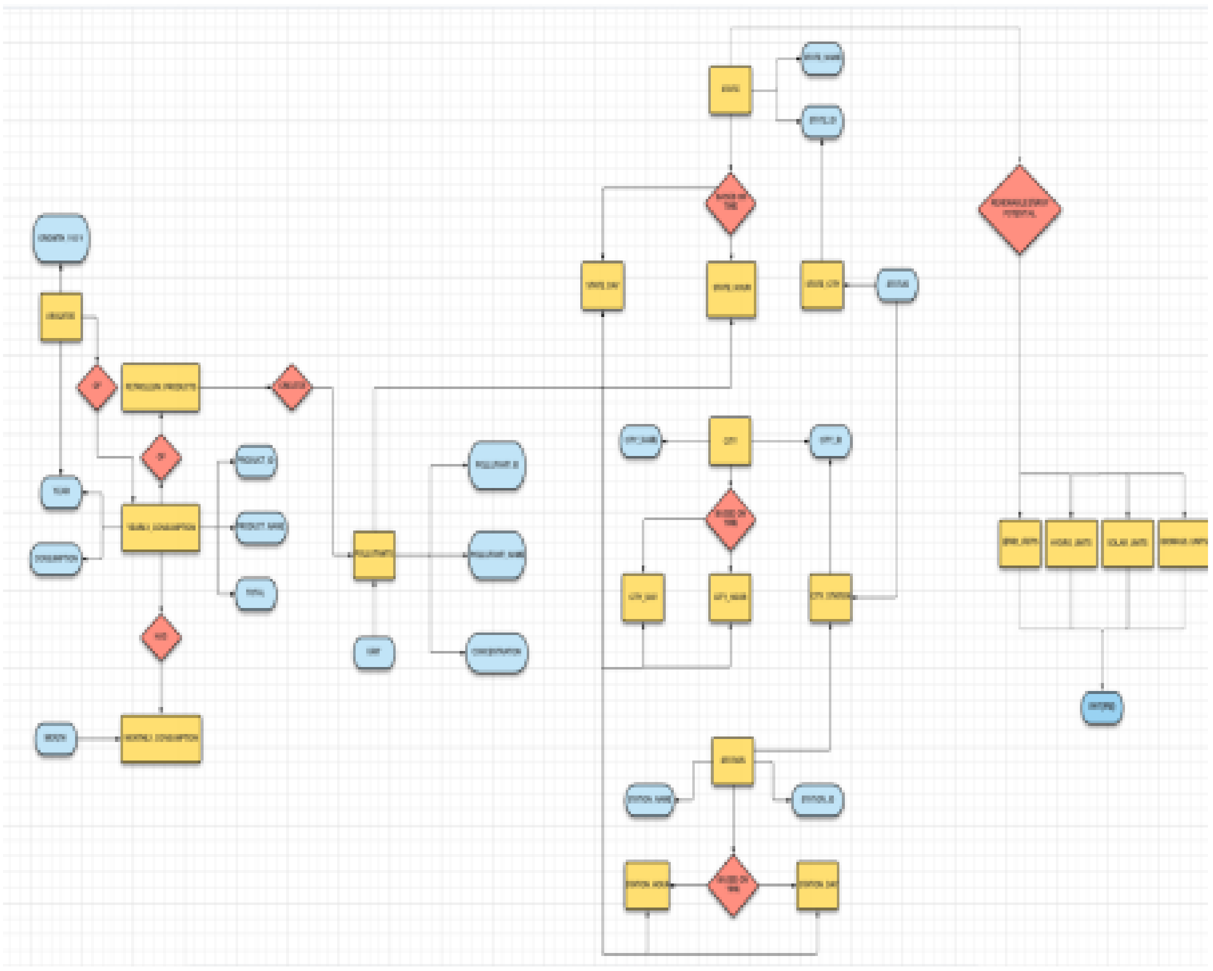


Table name	Super key	Candidate key	Primary key
Station	Station_Id, Station_name, [Station_Id, station name]	Station_ID,State _n ame	Station_ID
City_Stat ion	Station_ID, [Station_ID, City_ID]	[Station_ID, City_ID], Station_ID	Station_ID
City	City_ID, City_name, [City_ID, City_name]	City_ID, City_name	City_ID
State	State_ID, State_name	State_ID, State_name	State_ID

State_ Cit y	City_ID, [State_ID, City_ID]	[State_ID, City_ID],City_ID	City_ID
State_ Da y	[Date, Pollutant_ID, State_ID] [Date, Pollutant_ID, State_ID,Concentration]	[Date, Pollutant_ID, State_ID]	[Date, State_ID,Pollutant_ID]
State_ Hour	[Date_Time, State_ID, Pollutant_ID] [Date_Time,State_ID,Polluta nt_ID,Concentration]	[Date_Time, State_ID, Pollutant_ID]	[Date_Time, State_ID, Pollutant_ID]
Pollutant	Pollutant_ID, Pollutant_name, [Pollutant_ID, Pollutant_name], [Pollutant_ID,Pollutant_na me, 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,Sta tio n_ID,concentration]	[Date_time,Stati on _ID, Pollutant_ID]	[Date_time,Station_ID, Pollutant_ID]
Station Day	[Date,Station_ID,Pollutant_ ID] , [Date,Station_ID,Pollutant_ ID, concentration]	[Date,Station_I D,P ollutant_ID]	[Date,Station_ID,Pollutant_ID]
City Hour	[Date_time, Pollutant_ID, City_ID] [Date_time,Pollutant_ID,Cit y_I D,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]
Petroleu m Products	Product_Name, Product_ID [Product_ID,Product_Na me]	Product_Name, Product_ID	Product_ID
Yearly Consu mp tion	[Product_ID, Year] [Product_ID,Year,Total]	[Product_ID, Year]	[Product_ID,Year]

Monthly Consu mp tion	Product_ID, Month, Year, [Product_ID, Year], [Month, Year, Product_ID] [Month, Year, Product_ID, Consumptiom]	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,units]	State_ID, State_name	State_ID
Biomass Units	State_ID, State_name, [State_ID, State_name] [State_ID,State_name,units]	State_ID, State_name	State_ID
Hydro Units	State_ID, State_name, [State_ID, State_name] [State_ID,State_name,units]	State_ID, State_name	State_ID
Solar Units	State_ID, State_name, [State_ID, State_name] [State_ID,State_name,units]	State_ID, State_name	State_ID

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 5

1) TABLE CREATIONS

```
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
);
```

2) 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;
```

3) 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;
```

4) 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;
```

5) 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);

6) 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))
```

7) 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);

8) 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;
```

9) 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;
```

10) 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));
```

11) ANY

1) To find the information of pollution production on daily basis of all the cities.

```
select *
from city_day
where city_day.city_id= any(select city_id
 from city
 where city_day.city_id=city.City_ID);
```

2) To find the names of the cities which concentration of pollution greater than 28.

```
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) To find the names of the states which concentration of pollution greater than 28.
```
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) To find all the daily information of all states which have concentration of pollutants greater than 50 on hourly basis.
```
select *
from state_day
where state_day.concentration=any(select
state_hour.concentration  from state_hour
 where state_hour.concentration>50);
```

5) To find all the hourly information of all states which have concentration of pollutants greater than 50 on daily basis.
```
 select *
 from state_hour
 where state_hour.concentration=any(select
state_day.concentration  from state_day
 where state_day.concentration>50);
```

12) ALL

1. To find **ALL** the states producing above average power from solar-energy.
```
SELECT
state from solar
where
solarpower>all(select avg(solarpower) from solar);
```
2. To find **ALL** the states producing below average power from hydro-energy.
```
SELECT
states from hydro
where
hydropower<all(select avg(hydropower) from hydro);
```

3. To find **ALL** the stationed of stations with city id 4.
```
SELECT
stationid from city_station
```

```
where
cityid=all(select cityid from city_station where cityid=4);
```

4. To find **ALL** the states which produce more power from biomass than the average power produced from hydro-energy.

```
SELECT
states,stateid from biomass
where
power>all(select avg(hydropower) from hydro);
```

5. To find **ALL** the states which produce solar-power more than the maximum power produced by both hydro and biomass energy.

```
SELECT
state,stateid from solar
where
solarpower>all(select max(hydropower) from hydro)
and
solarpower>all(select max(power) from biomass);
```

ASSIGNMENT 6

1. STORED FUNCTIONS

Ø What are stored functions?

In MySQL, a stored function is a collection of SQL statements that perform a single task or operation and return a single value. In MySQL, it is one of the kinds of stored programmes. Make sure you have the CREATE ROUTINE database privilege before creating a stored function. This function was typically used to encapsulate reusable business rules or formulas in stored programmes or SQL statements.

The stored function is nearly identical to the process in MySQL, but there are a few exceptions like the function parameter may only have the IN parameter, but it cannot be specified, while the procedure may have IN, OUT, and INOUT parameters. Only one value specified in the function header can be returned by the stored function. The stored function can also be called from inside the programme.

```
CREATE FUNCTION `desnity_to_ppm` (conc decimal,pollutant varchar(10))

RETURNS decimal deterministic

BEGIN case
```

```
when pollutant='CO' then return conc*1000 ;

when pollutant='NOx' then return conc*0.001;

else return conc;

end case;

END$$

DELIMITER;


select *,desnity_to_ppm(city_day.concentration,city_day.pollutant) from city_day;
```

2. TUPULE RELATIONAL CALCULUS

1) To find the dates when cities produced pollutants with concentration more than 100.

   {c.date | city_day(c) ^ c.concentration ≥ 100}

2) To find the cities(city-id) which have their stations producing pollutants more than 100 on daily basis.

{c.cityid |(city_station(c) ∃ station_day(s)) ^ c.stationid = s.stationid ^s.concentration ≥ 100}

3) To display amount of concentration form city_hour

   {c.concentraion | city_hour(c) }

4) To display hours when concentration level was more or equal to 11.

   p.hour| city_hour(c) ^ c.concentration >11}

5) To display city hours when concentration is 10

   {c.hour | city_hour(c) ^ c.concentration ='10'}

6) To find states, states id and amount of biomass power produced greater than or

equal to 1000.

{t| t € biomass ∧ t[biomass power] >= 1000}

7) To find what amount of units produced in one day by each state.

{t| ∃ s ∈ hydro (t[units] = s[units] ∧ ∃ u ∈ solar (t[units] = u[units]) }

3. DOMAIN RELATIONAL CALCULUS

1) To find states which generate wind power greater than or equal to 10000.

{<s, p> | <s, p> ∈ wind ∧ (p >= 10000)}

2) To find yearly and monthly consumption of petroleum products from their id.

{<pi, mc, yc> | ∃ <pi, mc> ∈ monthly consumption ∇ (pi) ∧ (<pi, yc> ∈ (yearly consumption ∇ (pi)))}

3) Find time, pollutant where concentration is greater than 20

{<pollutant,concentration>/∈city_hour ^ concentration>20}

4) To find concentration when the datetime is 16-06-2017 14:00

{<concentration>/∈city_hour ^ datetime=` 16-06-2017 14:00 `}

5) To find the state-name and state-id of the state which produces solar energy more than 1 lakh KWh.

{<s,sid,p> |<s,sid,p>∈solar ^ p>1000000 }

6) To display state id and city id of cities of the corresponding states producing pollutants more than 100 concentration on the daily basis.

{<sid,cid> | ∃ cid (<sid,cid> ∈ state_city ^ ∃ c (<cid,d,p,c> ∈ city_day ^ (c ≥ 100))}

7) To find the name of the petroleum products with yearly consumption less than 10000.

{<pn, pi, c> | ∃ c (<pn, pi> ∊ Petroleum_Products ∧ ∃ c (<pi, y, c> ∊ Yearly_Consumption ∧ (c ≤ 10000)}

ASSIGNMENT 7 [MINIMAL COVER]

	Table Name	Functional Dependency	Minimal Cover	BCNF
1.	Station	Station Id -> Station_Name Station_Name-> Station ID	Station Id -> Station_Name	YES
2.	City	City Id -> City_Name City_Name-> City ID	City Id -> City_Name	YES
3.	State	State Id -> State_Name State_Name-> State ID	State Id -> State_Name	YES
4.	City_Station	Station ID -> City_ID	Station ID -> City_ID	YES
5.	State_City	City ID -> State_ID	City ID -> State_ID	YES
6.	Pollutant	Pollutant ID -> Pollutant_name, Unit Pollutant_Name -> Pollutant ID, Unit	Pollutant ID -> Pollutant_name Pollutant ID -> Unit	NO
7.	Solar	State_ID -> State_name, units State_name -> State_ID, units	State_ID - > State_name State_ID -> units	NO
8.	Wind	State_ID -> State_name, units State_name -> State_ID, units	State_ID - > State_name State_ID -> units	NO
9.	Hydro	State_ID -> State_name, units State_name -> State_ID, units	State_ID - > State_name State_ID -> units	NO
10	. Biomass	State_ID -> State_name, units State_name -> State_ID, units	State_ID - > State_name State_ID -> units	NO
11	. Station_hour	Station ID, Date_Time, Pollutant ID -> Concentration	Station ID, Date_Time, Pollutant ID -> Concentration	YES
12	. City_Hour	City_ID, Date_Time, Pollutant ID -> Concentration	City_ID, Date_Time, Pollutant ID -> Concentration	YES

13	. State_hour	State_ID, Date_Time, Pollutant ID -> Concentration	State_ID, Date_Time, Pollutant ID -> Concentration	YES

14.	Station_Day	Station ID, Date, Pollutant ID -> Concentration	Station ID, Date, Pollutant ID -> Concentration	YES
15.	City_Day	City_ID, Date, Pollutant ID -> Concentration	City_ID, Date, Pollutant ID -> Concentration	YES
16.	State_Day	State_ID, Date, Pollutant ID -> Concentration	State_ID, Date, Pollutant ID -> Concentration	YES
17.	Petroleum_Products	Product_Name -> Product_ID Product_ID-> Product_name	Product_ID-> Product_name	YES
18.	Monthly Consumption of Products	Product_ID, Month, Year -> Consumption	Product_ID, Month, Year -> Consumption	YES
19	. Yearly Consumption	Product_ID, Year -> Consumption	Product_ID, Year -> Consumption	YES
20	. Analysis	Year -> Y_O_Y Growth	Year -> Y_O_Y Growth	YES

Note : The tables are already in 1NF,2NF and 3NF form.