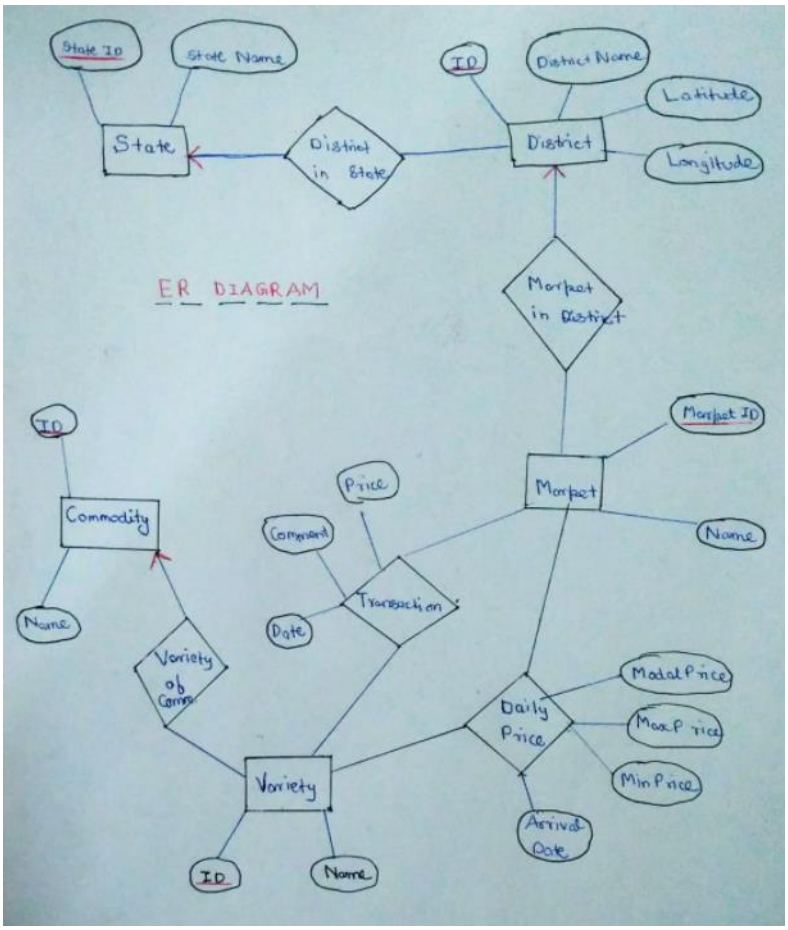
Agricultural Marketing Dashboard

Mohammed Ashraf (2017CS10589), Mayur Solanki (2017CS10349),
Vedant Raval (2017CS10366)

Section 1: Description

- We have implemented a system that would make use of the Information about the prices of the Agricultural Commodities in markets across different regions in India and provide certain meaningful insights along the following aspects:
  1. Price trends for various commodities across these different regions and markets so that the user can infer the trends and make use of the speculated predictions from these trends while making a decision about farming certain crops, selling them in certain markets etc.
  2. Optimal locations for the user to sell the commodity grown by them based on the price trends and distance based costs, so that the user can get the maximum profit for the product grown.
- Thus because of such functionalities of this system, it would be extremely useful for the users to get useful insights that would influence their decisions taken during the entire process of farming the crops to selling them.
- The ER diagram for the system can be found as:



- The different Entities and Attributes (Primary key in red font):
  - Table **state**

state_id: serial	state_name: text

  - Table **district**

district_id: serial	state_id: int	district_name: text
latitude: real	longitude: real	-

  - Table **market**

market_id: serial	district_id: int	market_name: text

  - Table **commodity**

commodity_id: serial	variety: text	commodity_name: text

  - Table **daily_prices**

record_id: bigserial	market_id: int	arrival_date: date	commodity_id: int
min_price: real	max_price: real	modal_price: real	-

  - Table **transaction_prices**

record_id: bigserial	market_id: int	transaction_date:date
price: real	commen: text	variety_id:int

Section 2: Getting the data

- We got the data corresponding the Agricultural market locations, prices etc. from *data.gov.in*. We combined that with the geographic data corresponding to the longitude and latitude of these different districts from *https://www.kaggle.com/sirpunch/indian-census-data-with-geospatial-indexing*.
- Thus using the links described above, we made use of readymade data
- The data that was obtained by us was not readymade and multiple files had to be collected and processed for use.

Section 2.1: Processing Steps:

- Conversion of some xml files to csv format
- Data was in a form like of a universal relation, so we had to process them for ease and for following good schema design practices.

- On account of above step, we processed over 30 files to convert information in our desired format.
- There were some other issues also, such as minor error in 'District' names in different files. For this, while processing, we managed to detect many common and most occurring cases and included steps for auto correction during the pre-processing time.

- **Size of the raw dataset (before clean-up):** 201MB
- **Size of the cleaned dataset:** 100 MB
- Stats for the different tables present in the dataset:

Table name	Number of Tuples	Time to load
state	37	1.585 ms
district	611	9.038 ms
market	2737	23.745 ms
commodity	8	1.166 ms
daily_prices	2849879 (2.8million)	81945.4 ms (81.9 s)

Section 3: View of the system

- **User view**
  - **Header**: Contains links for *Home page, Search page, Market Recommendations page, Add Entries,* and a page where we describe more *About us.* The header is present in all the pages for easy access
  - **Home Page**: Consists of three links in the body. One that goes to the *Search page,* one to the *Market Suggestions page,* and one to the *Add Entries* page.
  - **Search Page:** Consists of three links in the body. One that goes to the *Search on Commodity,* one to the *Search on Year,* and one to the *Search on location* page.
    - **Search on Commodity:** The user selects the state, district, and commodity from three different dropdowns and gets trends corresponding to them
    - **Search on Year:**
    - **Search on Location:**
  - **Market Recommendations page:** The user selects the state, district, commodity he's interested in, variety of that, and month of the year and based on that the systems gives out recommendations of the market to the user where he will get the best price for his crops

- **Add Entries:** This page allows the user to add price information to our database. This page includes fields of location information, market name, commodity, arrival data and price.

- **System view**
  - (Explain the functionality of Indexes etc in 2-3 lines)
  - (Make a table for the codes of all the SQL queries and the webpage that they correspond to)
- **List of the queries tested**

  Queries that may be executed while using the application:

Simple Queries for UI interfaces

1. Selecting list of states

```
select state_id, state_name
from state
order by state_name;
```

2. Selecting districts given a state

```
select d.district_id, d.district_name
from (select * from state where state_name = "STATE NAME") s
inner join district d on s.state_id = d.state_id
order by d.district_name;
```

3. Getting list of commodities

```
select c.commodity_id, v.variety_id, c.commodity_name, v.variety_name
from commmodity c
inner join variety v on variety.commodity_id = c.commodity_id
order by c.commodity_name, v.variety_name;
```

4. List of markets given state

```
select market.market_id, market.market_name from market
inner join district on market.district_id = district.district_id
inner join state on state.state_id = district.state_id
where state_name = '" + state_name + "';
```

5. List of markets given state and district

```
select market.market_id, market.market_name from market
inner join district on market.district_id = district.district_id and district.district_id
        = '" + district_id + "'
inner join state on state.state_id = district.state_id
where state_name = '" + state_name + "';
```

Queries for Result

6. Getting year wise trends for the given district and commodity

```
select d.year :: int , round(max (d.max _price) :: numeric, 2) :: int, round(min (d.min _price)
                :: numeric, 2) :: int ,       round(avg(d.modal_price)::numeric, 2)::int
from (select * from state where state_name = %s) st
inner join (select * from district where district_id = %s) dt on dt.state_id
                = st.state_id
inner join market m on m.district_id
                = dt.district_id
            inner join (select record_id, market_id, variety_id, extract (year from arrival_date)
as year, min_price, max_price, modal_price from daily_prices
        where min_price > 0 and max_price > 0 and modal_price > 0) d on d.market_id
                = m.market_id
inner join variety v on v.variety_id = d.variety_id
inner join (select * from commodity where commodity_name = %s) c on c.commodity_id
                = v.commodity_id
            group by d.year
order by d.year;
```

7. Year wise trends across all varieties at a location

```
with d as (
  select record_id, market_id, daily_prices.variety_id, extract (year from arrival_date)
as year, min_price, max_price, modal_price   from daily_prices
    11.   inner join variety v on v.variety_id = daily_prices.variety_id
  inner join commodity on commodity.commodity_id
                = v.commodity_id and commodity.commodity_name = %s
  where market_id in (select market.market_id from market
                where market.district_id = %s)
            and min_price > 0 and max_price > 0 and modal_price > 0
        )
select d.year :: int , round(max (d.max _price) :: numeric, 2) :: int,
round(min(d.min_price)::numeric, 2)::int , round(avg(d.modal_price)::numeric, 2)::int
from d
group by d.year
order by d.year;
```

8. Year wise trends for given variety and location

```
with d as (
        select record_id, market_id, daily_prices.variety_id, extract (year from arrival_date)as year,
min_price, max_price, modal_price
    from daily_prices
    inner join variety v on v.variety_id = daily_prices.variety_id and v.variety_name = %s
```

```
    inner join commodity on commodity.commodity_id
            = v.commodity_id and commodity.commodity_name = %s
    where market_id in (select market.market_id from market
        where market.district_id = %s)
    and min_price > 0 and max_price > 0 and modal_price > 0
)
select d.year :: int , round(max (d.max _price) :: numeric, 2) :: int,
round(min(d.min_price)::numeric, 2)::int , round(avg(d.modal_price)::numeric, 2)::int
from d
group by d.year
order by d.year;
```

9. Trends for a given commodity across different places

```
select st.state_name, dt.district_name, d.year :: int, round(max (d.max _price) :: numeric, 2)
            :: int,
round(min(d.min_price)::numeric, 2)::int , round(avg(d.modal_price)::numeric, 2)::int
  from state st
inner join district dt on dt.state_id = st.state_id
inner join market m on m.district_id = dt.district_id
inner join (select record_id, market_id, variety_id, extract (year from arrival_date) as year,
min_price, max_price, modal_price from daily_prices
  where min_price > 0 and max_price > 0 and modal_price > 0) d on d.market_id
            = m.market_id
inner join variety v on v.variety_id = d.variety_id
inner join (select * from commodity where commodity_name = %s) c on c.commodity_id
            = v.commodity_id
group by st.state_name, dt.district_name, d.year
order by st.state_name, dt.district_name, d.year;
```

10. Year wise trends for all districts for given commodities in a state

```
select st.state_name, dt.district_name, d.year::int, round(max(d.max_price)::numeric, 2)::int, round(min(
from state st
inner join district dt on dt.state_id = st.state_id
inner join market m on m.district_id = dt.district_id
inner join (select record_id, market_id, variety_id, extract (year from arrival_date) as year, min_price, max_
    where min_price > 0 and max_price > 0 and modal_price > 0) d
    on d.market_id = m.market_id
inner join variety v on v.variety_id = d.variety_id
inner join (select * from commodity where commodity_name = %s) c on c.commodity_id
            = v.commodity_id
where st.state_name = %s
group by st.state_name, dt.district_name, d.year
order by st.state_name, dt.district_name, d.year;
```

12. Getting the market recommendations across all varieties of a commodity.

```
with net_prices as (
```

```
  select dis.d2_id as district_id,round(avg(pt.modal_price_avg) − 0.6
            ∗ avg(dis.dist))::int as net_price,avg(dis.dist)::int as dist
  from (select ∗ from district where district_id = %s) d
  inner join state s on d.state_id = s.state_id and s.state_name = %s
  inner join distances dis on dis.d1_id = d.district_id and dis.dist < 500
  inner join market m on dis.d2_id = m.district_id
  inner join price_trend pt on m.market_id
= pt.market_id and pt.variety_id in (select variety_id from commodity,variety where commodity.commodi
= variety.commodity_id and commodity_name = %s and variety_name = %s)and pt.month
= %s
  group by dis.d2_id
)
select np.∗,d.district_name,s.state_name
from (select ∗ from net_prices order by net_price desc limit 10) np
inner join district d on d.district_id = np.district_id
inner join state s on d.state_id = s.state_id
order by np.net_price desc;
```

13. Getting the market recommendations for given variety of a commodity

```
with net_prices as (
  select dis.d2_id as district_id,round(avg(pt.modal_price_avg) − 0.6
            ∗ avg(dis.dist))::int as net_price,avg(dis.dist)::int as dist
  from (select ∗ from district where district_id = %s) d
  inner join state s on d.state_id = s.state_id and s.state_name = %s
  inner join distances dis on dis.d1_id = d.district_id and dis.dist < 500
  inner join market m on dis.d2_id = m.district_id
  inner join price_trend pt on m.market_id
= pt.market_id and pt.variety_id in (select variety_id from commodity,variety where commodity.commodi
= variety.commodity_id and commodity_name = %s)and pt.month = %s
  group by dis.d2_id
  )
select np.∗,d.district_name,s.state_name
from (select ∗ from net_prices order by net_price desc limit 10) np
inner join district d on d.district_id = np.district_id
inner join state s on d.state_id = s.state_id
order by np.net_price desc;
```

14. Inserting a record in the transaction price table

```
insert into transaction_prices(market_id,transaction_date,price,variety_id,comment) values
(%s,%s,%s,%s,%s);
```

Queries Used In Triggers

15. Adding new record to daily prices due to new addition in transaction prices

```
insert into daily_prices(market_id,arrival_date,commodity_id,min_price,max_price,modal_price)
values (NEW.market_id,NEW.transaction_date,NEW.commodity_id,NEW.price
```

$, NEW.price, NEW.price);$

16. Updating the values in daily prices while responding to trigger

i)

```
update daily_prices
set max_price = NEW.price
where market_id = NEW.market_id and arrival_date NEW.transaction_date and
  commodity_id = NEW.commodity_id and max_price < NEW.price;
```

ii)

```
update daily_prices
set min_price = NEW.price
where market_id = NEW.market_id and arrival_date = NEW.transaction_date and
commodity_id = NEW.commodity_id and min_price > NEW.price;
```

iii)

```
update daily_prices
set modal_price = (
  with m as (select price, count(*) as buys from
    transaction_prices
    where market_id = NEW.market_id and arrival_date
          = NEW.transaction_date and commodity_id = NEW.commodity_id
    group by price
  )
  select avg(m.price) from m
  where buys = (select max(buys) from m)
    )
where market_id = NEW.market_id and arrival_date
          = NEW.transaction_date and commodity_id = NEW.commodity_id;
```

Section 4. Tested Queries

- Query No: 6 | Time taken: 264 ms
  - state_name: Delhi
  - district_id: 121
  - commodity_name: Tomato
- Query No: 7 | Time taken: 260 ms
  - district_id: 254
  - commodity_name: Onion
- Query No: 8 | Time taken: 252 ms
  - district_id: 121
  - commodity_name: Tomato

- Query No: 10 | Time taken: 370 ms
  - state_name: Uttar Pradesh
  - commodity_name: Wheat

- Query No: 12 | Time taken: 88 ms
  - state_name: Delhi
  - district_id: 121
  - commodity_name: Tomato
  - month: 9