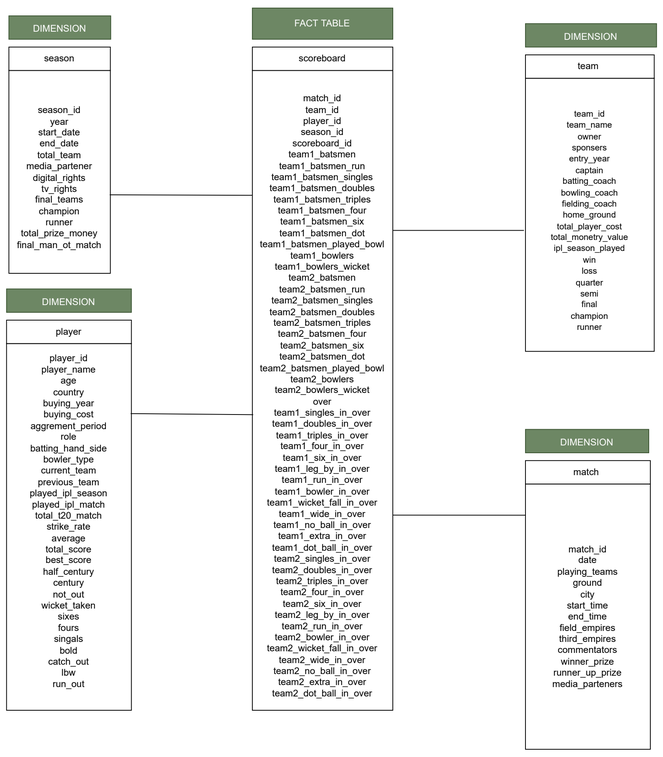
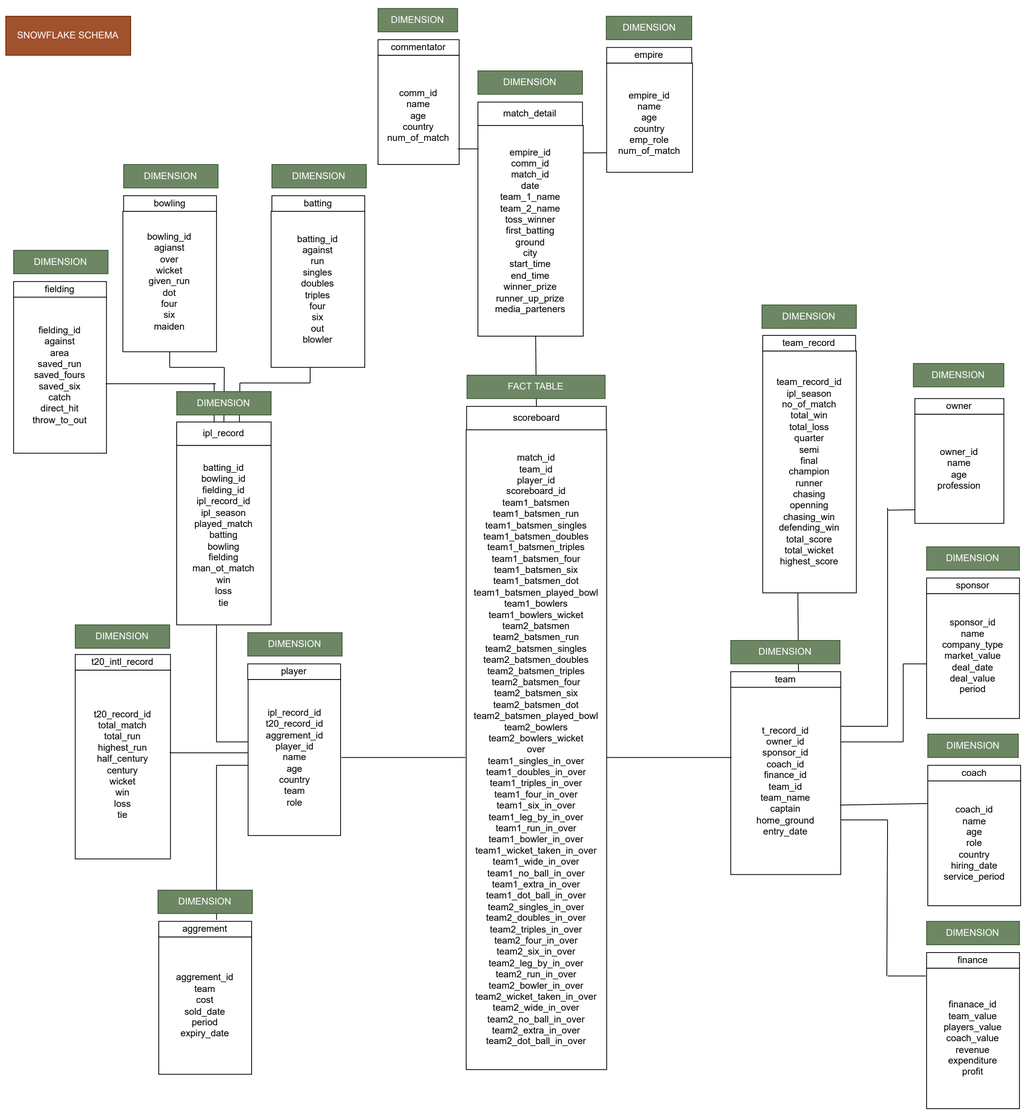
1	IPL TOURNAMENT DATA WAREHOUSE

STAR SCHEMA



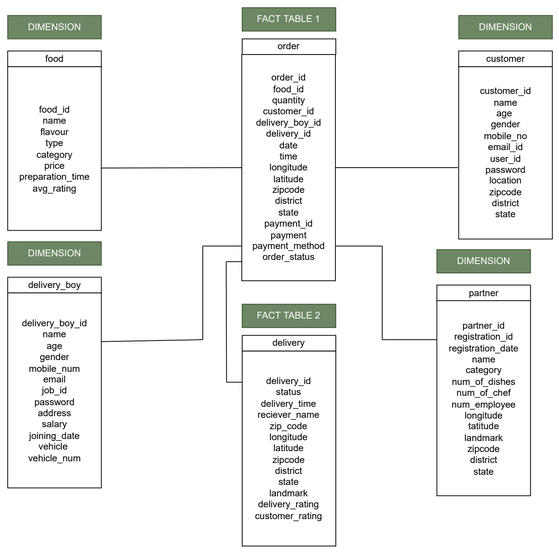
ABOUT DATAWAREHOSE - *This IPL datawarehose has one fact table score which consissts every possible aspect of match palyed by both team. We can get every important information and facts about match.* *There more than one foreign key to precisely get data from table, for example, team1_batsman and team2_batsman both are foreign key so we can directly join with team table and get records.* *The fact table has three dimensions, match, player, team to get additional information about match.*	
SQL COMMONDS ON TALBES - 1 - To know both team total score. SELECT SUM(team1_run_in_over) AS team_1_total_run, SUM(team2_run_in_over) AS team_2_total_run FROM scoreboard; 2 - To know team 1 batsmen performance. SELECT player_id, player_name, SUM(team1_batsmen_run) FROM scoreboard INNER JOIN player ON scoreboard.team1_batsmen = player.player_id GROUP BY player_id; 3 - Fall of wickets of both teams in overs. SELECT over, team1_wicket_fall_in_over, team2_wicket_fall_in_over FROM scoreboard;	4 - Wicket taken by team 2 bowlers. SELECT player_name AS team1_bowler, SUM(team1_wicket_fall_in_over) AS wickets FROM scoreboard INNER JOIN player ON scoreboard.team2_bowler_in_over = player.player_id GROUP BY team2_bowler_in_over; 5 - Maiden over by both team. SELECT team1_run_in_over AS team2_maiden, team2_run_in_one_over AS team1_maiden FROM scoreboard WHERE team1_run_in_over = 0 AND team2_run_in_over = 0; 6 - Century by team 1 batsmen. SELECT batsmen, total_run FROM (SELECT player_name AS batsman, SUM(team1_run_in_over) as total_run FROM scoreboard INNER JOIN player ON scoreboard.team1_batsmen = player.player_id GROUP BY team1_batsmen) x WHERE total_run > 99;

SNOWFLAKE SCHEMA



About Data Warehouse - *The data warehouse of IPL Tournament consists fact table and its three dimensions with their own dimensions lookup tables.* *The fact table 'scoreboard' is designed to do thorough analysis to the match played by two teams and their performances. It has 'player', 'team', 'match_schedule' dimensions to know additional information to get better understanding of facts.* *In fact table columns names are bit longer to give clear idea about the specific facts of performace from different perspective as much as possible in easy way, so we can run sql command to get the very detail we want.*	
SQL COMMAND ON TABLES 1 - To know teams names, who won the toss and who bat first SELECT team_1_name, team_2_name, toss_winner, first_batting FROM scoreboard INNER JOIN match_detail ON scoreboard.match_id = match_detail.match_id; 2 - To know teams total run in match SELECT SUM(team_1) AS team1_total_run, SUM(team2_batsmen_run) AS team2_total_run FROM scoreboard; 3 - To know how much run both teams made in first five overs SELECT over, team1_run_in_over, team2_run_in_over FROM scoreboard LIMIT 5; 4 - To know each batsman run SELECT team1_batsmen_run, team2_batsmen_run FROM scoreboard; 5 - Most run by batsmen in both team SELECT team_1_batsmen_run, team2_batsmen_run FROM scoreboard ORDER BY team1_batsmen_run DESC, team2_batsmen_run DESC; 6 - Run by batsmen in team 1 SELECT name AS batsmen_name, team1_batsmen_run AS run FROM scoreboard INNER JOIN player ON scoreboard.team1_batsmen = player.player_id ORDER BY team1_batsmen_run DESC; 6 - Run by batsmen in team 2 SELECT name AS batsmen_name, team2_batsmen_run AS run FROM scoreboard INNER JOIN player ON scoreboard.team2_batsmen = player.player_id ORDER BY team2_batsmen_run DESC; 8 - How many runs teams made by singles SELECT SUM(team1_singles_in_over), SUM(team2_singles_in_over) FROM scoreboard; 9 - How many fours and sixes both teams hit SELECT COUNT(team_1_four_in_over), COUNT(team1_six_in_over), COUNT(team2_four_in_over), COUNT(team2_six_in_over) FROM scoreboard; 10 - Knowing run rate of innings of both teams SELECT SUM(team1_run_in_over) / 20 AS run_rate_team1, SUM(team2_run_in_over) / 20 AS run_rate_team2 FROM scoreboard;	11 - To know team 1 player previous records SELECT name, total_match, win, loss, tie FROM scoreboard INNER JOIN player ON scoreboard.team1_batsmen = player.player_id INNER JOIN ipl_record ON player.player_id = ipl_record.ipl_record_id; 12 - To know batting record of team 2 in previous ipl matches SEELCT name AS team1_batsman, ipl_season, against, run FROM scoreboard INNER JOIN player ON scoreboard.team2_batsmen = player.player_id INNER JOIN ipl_record ON player.ipl_record_id = ipl_record.ipl_record_id INNER JOIN batting ON ipl_record.batting = batting.batting_id 13 - To know teams playing records in ipl SELECT team_name, ipl_season, no_of_match, total_win_total_loss, champion FROM scoreboard INNER JOIN team ON scoreboard.team_id = team.team_id INNER JOIN team_record ON team.team_record_id = team_record.team_record_id 14 - To know about finance of teams SELECT team_name, team_value, players_value,coach_value, revenue, expenditure, profit FROM scoreboard INNER JOIN team ON scoreboard.team_id = team.team_id INNER JOIN finance ON team.finance_id = finance.finance_id 15 - To know current match details SELECT date, team1_name, team1_name, toss_winner, first_batting, empire.name, commentator.name, ground FROM scoreboard INNER JOIN match_detail ON scoreboard.match_id = match_detail.match_id INNER JOIN empire ON match_detail.empire_id = empire.empire_id INNER JOIN commentator ON match_detail.comm_id = commentator.comm_id 16 -

STAR SCHEMA



ABOUT DATAWAREHOSE -
This is start schema of food delivery datawarehosue. This contains two fact table 'order' and 'delivery' with four dimensions.
we can get information about order and status to delivery and its status and order location to delivery location and date, time payment method and amout, and for additional information we can use their dimensions.

1 - To know food order, quantity, date and delivery status.

```
SELECT food_id, food_name, quantity, delivery.status FROM order
INNER JOIN food
ON order.food_id = food.food_id;
```

2 - To know which gender is ordering food more these days based on one year trend.

```
SELECT gender, COUNT(order_id) AS total_order FROM order
INNER JOIN customer
ON order.customer_id = customer.customer_id
GROUP BY gender;
```

3 - In what state people are ordering Frech Fries more based on previous month record.

```
SELECT state, COUNT(order_id) AS total_order FROM order
INNER JOIN food
ON order.food_id = food.food_id
WHERE food_name = 'French Fries'
GROUP BY state;
```

4 - Average rating last year.

```
SELECT avg(delivery_rating) AS service_rating FROM order
INNER JOIN delivery
ON order.delivery_id = delivery.delivery_id;
```

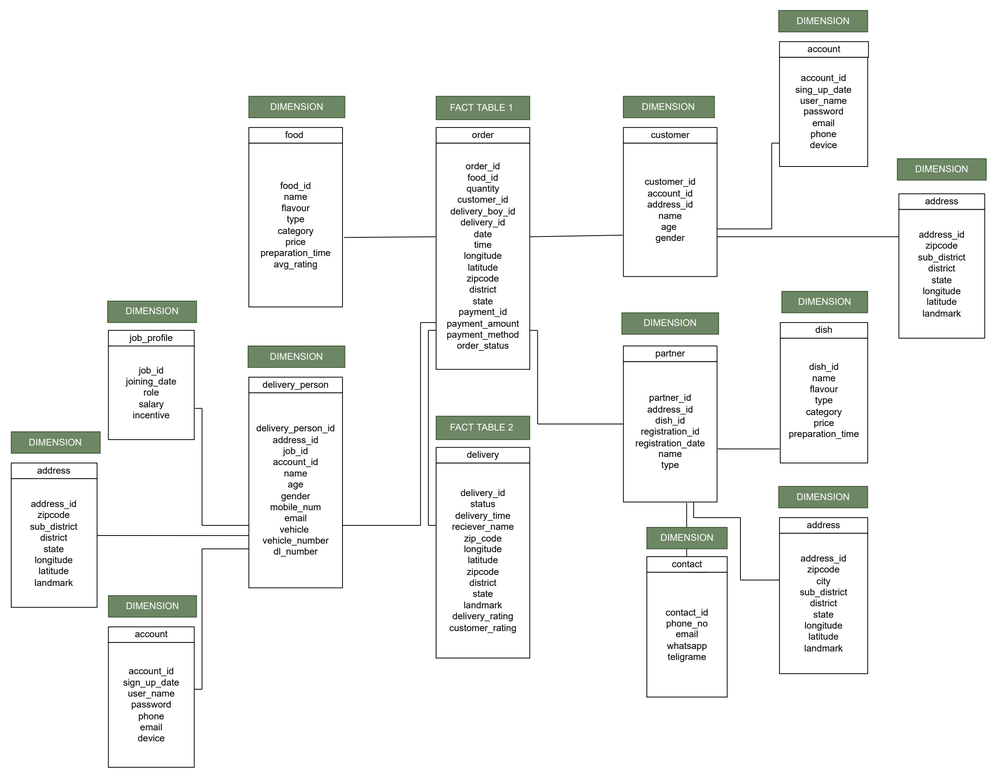
5 - Which state has more customer base.

```
SELECT state, COUNT(customer_id) AS total_customer FROM order
INNER JOIN customer
ON order.customer_id = customer.customer_id
GROUP BY state;
```

6 - Most rated food.

```
SELECT food_id, food_name, avg_rating FROM order
INNER JOIN food
ON order.food_id = food.food_id
ORDER BY DESC avg_rating;
```

SNOWFLAKE SCHEMA



ABOUT DATA WAREHOUSE -

The design consists two fact table 'order' and 'delivery' with four more dimension with their dimensions.
We can get information about the order, customer, delivery, payment, food, delivery person and the location of order and delivery. It has dimension of partners so we can get idea of who is supplying food and their food variety.

SQL COMMONDS ON TABLES

1 - To know order, food, date, time and delivery time, payment

```
SELECT date, time, order_id,  name AS food, quantity, payment, order.status, delivery_id, delivery.status FROM order
INNER JOIN delivery
ON order.delivery_id = delivery.delivery_id
INNER JOIN food
ON order.food_id = food.food_id;
```

2 - To know how many order in particular month.

```
SELECT COUNT(order_id) AS total_order_in_month FROM order WHERE date BETWEEN '2023-01-01' AND '2020-01-31';
```

3 - To know how much amount of order recieved in on year;

```
SELECT SUM(payment_amount) AS total_amount_in_year FROM order WHERE date LIKE  '%2022%';
```

4 - To know state wise total order in year.

```
SELECT COUNT(order_id) AS total_order FROM order WHERE date LIKE '%2022%';
```

5 - State wise total order in year.

```
SELECT state, COUNT(order_id) AS total_order_in_2018 FROM orders WHERE date LIKE '%2018%'
GROUP BY state;
```

6 - Mostly used payment method in last 5 year.

```
SELECT payment_method, COUNT(payment_method) AS totaly_used_in_5_year
FROM oreder WHERE date BETWEEN '2017-01-01' AND '2022-12-31' GROUP BY payment_method;
```

7 - Age group orders in one year.

```
SELECT age, count(order_id) as total_order FROM order
INNER JOIN customer
ON order.customer_id = customer.customer_id
GROUP BY age;
```

8 - Highly order foods under 30 age group in last year.

```
SELECT food.name, COUNT(order_id) FROM order
INNER JOIN customer
ON order.customer_id = customer.customer_id
INNER JOIN food
ON order.food_id = food.food_id
WHERE date <= 30
GROUP BY food.name;
```

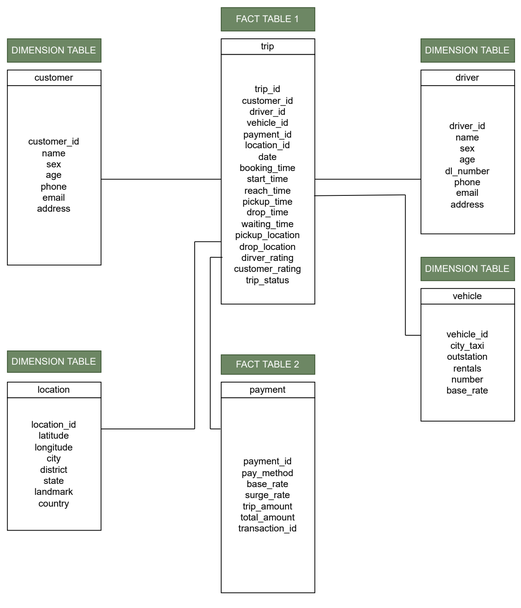
9 - To know in which state people more ordering vegeterian food.

```
SELECT state, COUNT(order_id) AS order FROM order
INNER JOIN food
ON order.food_id = food.food_id
WHERE food.category = 'veg'
GROUP BY state;
```

10 - Average salary of delivery person

```
SELECT avg(salary) AS average_salary FROM order
INNER JOIN delivery_person
ON order.delivery_person_id = delivery_person.delivery_person_id
INNER JOIN job_profile
ON delivery_person.job_id = job_profile.job_id;
```


STAR SCHEMA



About datawarehouse -
This is start schema consisting four dimensions, location, vehicle, customer, driver.
The schema has two fact table, trip and payment.

SQL COMMONDS ON TABLES -

1 - To know how many trip happended in last year.

```
SELECT COUNT(trip_id) AS total_trip_in_2022 FROM trip WHERE date BETWEEN '2022-01-01' AND '2022-12-31';
```

2 - To know statewise trip data.

```
SELECT state, COUNT(trip_id) FROM trip
INNER JOIN location
ON trip.location_id = location.location_id
GROUP BY state;
```

3 - How many trip have been cancelled and how many not cancelled.

```
SELECT COUNT(trip_status) AS not_concelled, (SELECT COUNT(trip_status) AS cancelled FROM trip WHERE trip_status
= 'ride_cancelled') FROM trip WHERE trip_status = 'ride_on';
```

4 - How much revenue generated in last year.

```
SELECT SUM(total_amount) AS total_revenue_in_2022 FROM trip;
```

5 - Most used vehicle in city ride.

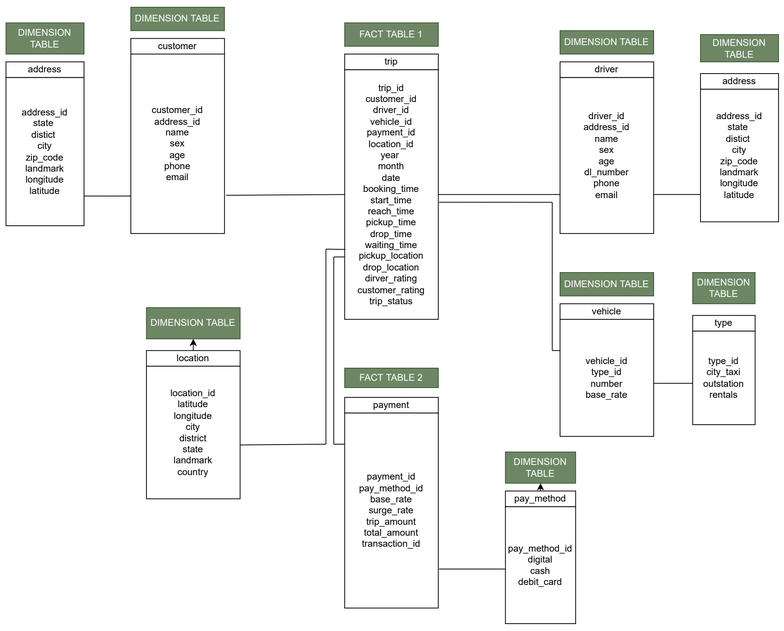
```
SELECT city_taxi, total_number FROM
(SELECT city_taxi, COUNT(city_taxi) AS total_number FROM trip
INNER JOIN vehicle
ON trip.vehicle_id = vehicle.vehicle_id
GROUP BY city_taxi) AS x
ORDER BY total_number DESC LIMIT 1;
```

6 - Total use of different payment methods.

```
SELECT payment_method, total_use_count FROM

(SELECT payment_method, COUNT(payment_method) total_use_count FROM trip
INNER JOIN payment
ON trip.payment_id = payment.payment_id
GROUP BY payment_method) AS x
ORDER BY total_use_count DESC;
```

SNOWFLAKE SCHEMA



About Data warehouse -
This is snowflake schema with two fact tables and several dimensions.
We can get many information regarding cab ride from pick up time and place to drop time and place, payment amount, method and vehical type, this data warehouse schema provides us to analyze the data.

1 - How many customer company has.

```
SELECT COUNT(customer_id) AS total_customers FROM trip
INNER JOIN customer
ON trip.customer_id = customer.customer_id;
```

2 - How many trip occurring in last year.

```
SELECT COUNT(trip_id) AS total_trip_2022 FROM trip
WHERE date BETWEEN  '2022-01-01' AND '2022-12-31';
```

3 - Average drivers rating.

```
SELECT AVG(driver_rating) AS average_dirver_rating FROM trip;
```

4 - Yearly revenue for last five year.

```
SELECT SUM(total_amount) AS total_revenue FROM trip
GROUP BY year WHERE year BETWEEN '2017' AND '2022';
```

5 - Monthly avg trip for last five year.

```
SELECT AVG(trip_id) AS averge_trip_monthly FROM trip
GROUP BY month WHERE year BETWEEN ' '2017' AND '2022';
```

6 - What kind of age group using services in what number last year.

```
SELECT customer.age AS age, COUNT(trip_id) AS total_trip_count FROM trip
INNER JOIN customer
ON trip.customer_id = customer.customer_id
GROUP BY customer.age WHER trip.year = 2022;
```

7 - In what city most revenue is comming from based on five year data.

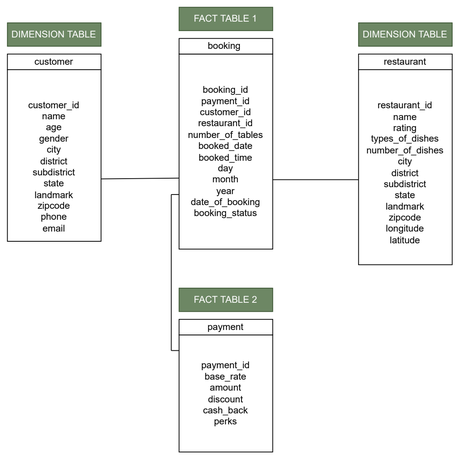
```
SELECT city, total_revenue FROM
(SELECT location.city AS city, SUM(total_amount) AS total_revenue FROM trip
INNER JOIN location
ON trip.location_id = location.location_id
INNER JOIN payment
ON trip.payment_id = payment.payment_id
GROUP BY city WHERE trip.year BETWEEN '2017' AND '2022') AS x
ORDER BY toal_revenue DESC LIMIT 1;
```

8 - Total female customer base.

```
SELECT COUNT(customer_id) AS total_female_customer FROM trip
INNER JOIN customer
ON trip.customer_id = customer.customer_id
WHERE customer.sex = 'female';
```

4	RESTAURANT TABLE BOOKING DATA WAREHOUSE

STAR SCHEMA



About Data Warehouse -
This is restaurant table booking data warehouse with two fact table and two dimension tables.
We can get information about booking time, number of table, date, payment amout, method, retaurant and its location and number of dishes they are serving.

1 - How many booking took place in last year.

```
SELECT COUNT(booking_id) AS total_booking_2022 FROM booking;
```

2 - How many percentage bookings are using app payment system.

```
SELECT (COUNT(payment_id) / COUNT(booking_id)) * 100 AS percentage_of_app_payment FROM booking
INNER JOIN payment
ON booking.payment_id = payment.payment_id;
```

3 - Average bookingn in evey month in last year.

```
SELECT month, AVG(booking_id) FROM booking
ORDER BY month WHERE year = '2022';
```

4 - Time pattern of booking and total count of them in last year.

```
SELECT booked_time, COUNT(booked_time) AS total_count FROM booking
GROUP BY booked_time;
```

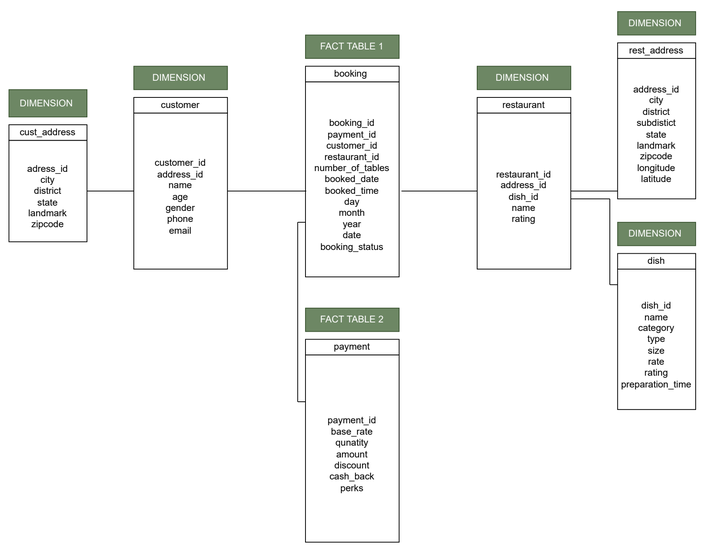
5 - State wise total customer base count.

```
SELECT state, COUNT(customer_id) AS total_customer FROM booking
INNER JOIN customer
ON booking.customer_id = customer.customer_id;
```

6 - Most booked restaurant list in Mumbai.

```
SELECT restaurant_id, restaurant.name, COUNT(booking_id) AS total_booked_number FROM booking
INNER JOIN restaurant
ON booking.restaurant_id = restaurant.restaurant_id
GROUP BY restaurant.name WHERE restaurant.city = 'Mumbai';
```

SNOWFLAKE SCHEMA



ABOUT DATA WAREHOUSE -
1 - This is snowflake schema for restaurant table booking data warehouse with two fact tables, booking and payment consisting other dimensions like customer, restaurant and theri own dimensins.
2 - It can give many valuable information regarding its business and services like booking number monthly, yearly and its time, payment total customer base and partener restaurant's data.

1 - Total booking year wise.

```
SELECT year, COUNT(booking_id) AS total_count FROM booking
GROUP  BY year;
```

2 - Total number of customer state wise.

```
SELECT cust_address.state, COUNT(customer_id) AS total_customer FROM
INNER JOIN customer
ON booking.customer_id = customer.customer_id
INNER JOIN cust_id
ON customer.address_id = cust_address.address_id
GROUP BY cust_address.state;
```

3 - Age group total booking list.

```
SELECT customer.age AS customer_age, COUNT(booking_id) AS total_booking_count FROM booking
INNER JOIN customer
ON booking.customer_id = customer.customer_id
GROUP BY customer.age;
```

4 - List of most booking in certain areas in Delhil

```
SELECT area, total_booking_count FROM
(SELECT rest_address.subdistrict AS area, COUNT(booking_id) AS total_booking_count FROM booking
INNER JOIN restauran
ON booking.restaurant_id = restaurant.restaurant_id
INNER JOIN rest_address
ON restaurant.resataurant_id = rest_address.address_id
GROUP BY rest_address.subdistrict WHERE rest_address.city = 'Delhi') AS x
ORDER BY total_booking_count DESC;
```

5 - What king of dishes restaurants are offering in Kolkata city.

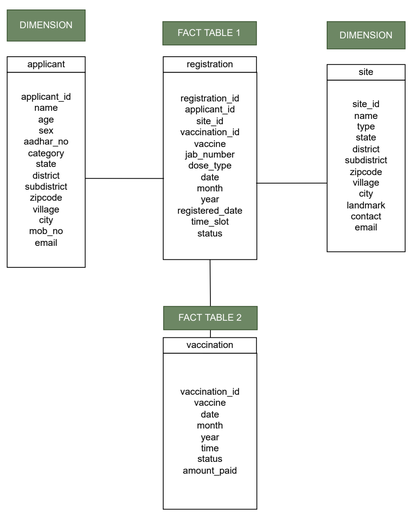
```
SELECT dish.dish_id, dish.name, dish.category AS category FROM booking
INNER JOIN restaurant
ON booking.restaurant_id = restaurant.restaurant_id
INNER JOIN dish
ON restaurant.dish_id = dish.dish_id;
```

6 - Gender based booking in last couple of years.

```
SELECT customer.gender AS gender, COUNT(booking_id) AS total_booking_count FROM booking
INNER JOIN customer
ON booking.customer_id = customer.customer_id
GROUP BY customer.gender WHERE booking.year BETWEEN '2019' AND '2022';
```

5 COVID VACCINATION APPLICATION DATA WAREHOUSE

STAR SCHEMA



ABOUT DATA WAREHOUSE -

1 - This data warehouse consists two fact table, registration and vaccination with two dimensionn tables, applicant and sites (of vaccination).
2 - We can get many important information by using these tables from registration of jab to vaccination completion and we can match both data to get any king of anamoly as well between vaccination name and status and dates.

1 - How many people registered for vaccine.

```
SELECT COUNT(registration_id) AS total_registration FROM registration;
```

2 - How many people got vaccinated yet.

```
SELECT COUNT(vaccination_id) AS total_vaccinated FROM registration
INNER JOIN vaccination
ON registration.vaccination_id = vaccination.vaccination_id;
```

3 - How many kind of vaccine is being given to people.

```
SELECT DISTINCT vaccine FROM registration.
```

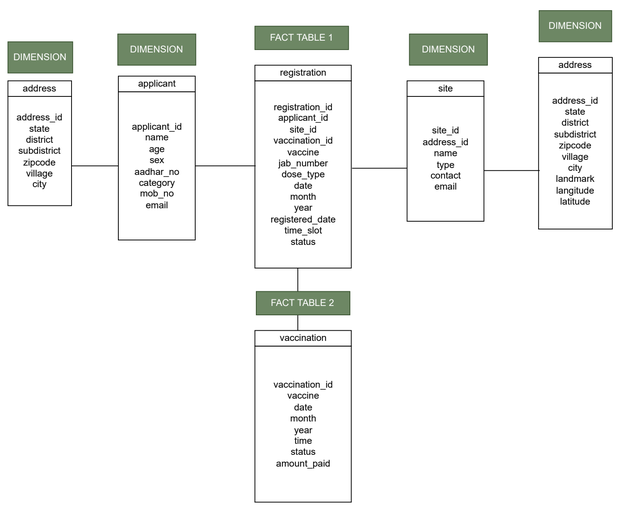
4 - How many people got vaccinated, different kind of vaccine wise total count.

```
SELECT vaccination.vaccine, COUNT(vaccination_id) FROM registration
INNER JOIN vaccination
ON registration.vaccination_id = vaccination.vaccination_id
GROUP BY vaccination.vaccine;
```

5 - State wise total vaccination data.

```
SELECT applicant.state, COUNT(vaccination_id) AS total_vaccinated_people
FROM registration
INNER JOIN vaccination
ON registration.vaccination_id = vaccination.vaccination_id
INNER JOIN applicant
ON registration.applicant_id = applicant.applicant.applicant_id
GROUP BY applicant.state;
```

SNOWFLAKE SCHEMA



ABOUT DATA WAREHOUSE -

1 - This is snowflake schema data warehouse consisting two fact tables, registration and vaccination with two more dimension with their dimensions.
2 - It can give multiple information about vaccination dirve around country state wise to district wise. We can get information about age group to how many jab the got and when and where, and many more.

1 - How many vaccination sites are running across the country.

```
SELECT COUNT(site_id) AS total_vaccination_sites FROM registration
INNER JOIN site
ON registration.site_id = site.site_id;
```

2 - How many government and non-government vaccination sites are available in country.

```
SELECT site.type, COUNT(site_id) FROM registration
INNER JOIN site
ON registration.site_id = site.site_id
GROUP BY site.type;
```

3 - Which state is most vaccinated.

```
SELECT state, total_vaccinated_people FROM
(SELECT applicant.state as state, COUNT(vaccinated_id) as total_vaccinated_people FROM registration
INNER JOIN applicant
ON registration.applicant_id = applicant.applicant_id
GROUP BY applicant.state) AS x
ORDER BY total_vaccinated_people DESC LIMIT 1;
```

4 - Total number of vaccinated people age wise.

```
SELECT applicant.age AS age, COUNT(vaccination_id) AS total_vaccinated_count FROM registration
INNER JOIN applicant
ON regisration.applicant_id = applicant.applicant_id
GROUP BY applicant.age;
```

5 - Average rate of vaccines in non-government hospitals (because in only private hospitals it was not free and have some different price in different places)

```
SELECT vaccination.vaccine, AVG(amount_paid) AS average_rate FROM registration
INNER JOIN vaccination
ON registration.vaccination_id = vaccination.vaccination_id
GROUP BY vaccination.vaccine;
```

6 - Total female vaccinated people.

```
SELECT COUNT(vaccination_id) AS total_female_vaccinated FROM registration
INNER JOIN applicant
ON registration.applicant_id = applicant.applicant_id
INNER JOIN vaccination
ON registration.vaccination_id = vaccination.vaccination_id
WHERE applicant.sex = 'female';
```