Design Data Warehouses For Given Below Products:
Note : While designing any Data Warehouse make sure to cover given below points.
a. Design Fact & Dimension tables
b. Create meaningful Primary & Foreign keys
c. Try to follow Star/SnowFlake Schema Design
d. Try to write few SQL queries to generate insightful business metrics (This is the critical point because you need to understand the Data & Business both)

1. Design a Data Warehouse for IPL Cricket Tournament **(Asked in Flipkart Interview for Senior Data Engineer role)**
Following is a data warehouse for IPL cricket tournament. Here is a possible schema:

Fact table: Matches
- match_id (PK): a unique identifier for each match
- season_id (FK): a foreign key referencing the Seasons dimension table
- venue_id (FK): a foreign key referencing the Venues dimension table
- team1_id (FK): a foreign key referencing the Teams dimension table, indicating the first team in the match
- team2_id (FK): a foreign key referencing the Teams dimension table, indicating the second team in the match
- toss_winner_id (FK): a foreign key referencing the Teams dimension table, indicating the team that won the toss
- toss_decision: a string indicating whether the toss winner chose to bat or field first
- result: a string indicating the outcome of the match, such as “normal”, “tie”, “no result”, etc.
- winner_id (FK): a foreign key referencing the Teams dimension table, indicating the team that won the match
- win_by_runs: an integer indicating the margin of victory in terms of runs, if applicable
- win_by_wickets: an integer indicating the margin of victory in terms of wickets, if applicable
- player_of_match_id (FK): a foreign key referencing the Players dimension table, indicating the player who was awarded the player of the match award

Dimension table: Seasons
- season_id (PK): a unique identifier for each season
- year: an integer indicating the year of the season
- start_date: a date indicating the start date of the season
- end_date: a date indicating the end date of the season

Dimension table: Venues
- venue_id (PK): a unique identifier for each venue
- name: a string indicating the name of the venue
- city: a string indicating the city where the venue is located
- country: a string indicating the country where the venue is located

Dimension table: Teams
- team_id (PK): a unique identifier for each team
- name: a string indicating the name of the team
- abbreviation: a string indicating the short form of the team name
- owner: a string indicating the owner of the team
- captain_id (FK): a foreign key referencing the Players dimension table, indicating the captain of the team

Dimension table: Players
- player_id (PK): a unique identifier for each player
- name: a string indicating the name of the player
- nationality: a string indicating the nationality of the player
- role: a string indicating the role of the player, such as “batsman”, “bowler”, “allrounder”, etc.

- batting_hand: a string indicating whether the player bats with left or right hand
- bowling_hand: a string indicating whether the player bowls with left or right hand
- bowling_skill: a string indicating the type of bowling skill of the player, such as “fast”, “medium”, “spin”, etc.

This schema follows a star schema design, as there is one fact table and multiple dimension tables that are directly connected to it. The fact table contains numerical measures and foreign keys, while the dimension tables contain descriptive attributes and primary keys.
Here are some example SQL queries to generate insightful business metrics from this data warehouse:

– Find out how many matches were played in each season and how many were won by each team

```
SELECT s.year, t.name AS team, COUNT(m.match_id) AS matches_played, SUM(CASE
WHEN m.winner_id = t.team_id THEN 1 ELSE 0 END) AS matches_won
FROM Matches m
JOIN Seasons s ON m.season_id = s.season_id
JOIN Teams t ON m.team1_id = t.team_id OR m.team2_id = t.team_id
GROUP BY s.year, t.name;
```

Copy

– Find out which venue hosted the most matches and which team had the best win percentage at that venue

```
WITH venue_matches AS (
  SELECT v.name AS venue, COUNT(m.match_id) AS matches_hosted
  FROM Matches m
  JOIN Venues v ON m.venue_id = v.venue_id
  GROUP BY v.name
  ORDER BY matches_hosted DESC
  LIMIT 1 -- get only one row with highest matches_hosted value
)

SELECT vm.venue, t.name AS team, COUNT(m.match_id) AS matches_played,
SUM(CASE WHEN m.winner_id = t.team_id THEN 1 ELSE 0 END) AS matches_won,
ROUND(100.0 * SUM(CASE WHEN m.winner_id = t.team_id THEN 1 ELSE 0 END) /
```

2. Design a Data Warehouse for Food delivery app like Swiggy, Zomato **(Asked in Grab for Data Engineer role)**

Here is a possible schema:

Fact table: Orders

- order_id (PK): a unique identifier for each order
- customer_id (FK): a foreign key referencing the Customers dimension table
- restaurant_id (FK): a foreign key referencing the Restaurants dimension table
- delivery_person_id (FK): a foreign key referencing the Delivery_Persons dimension table
- order_date: a date indicating the date of the order
- order_time: a time indicating the time of the order
- order_status: a string indicating the status of the order, such as “placed”, “confirmed”, “dispatched”, “delivered”, etc.
- total_amount: a decimal indicating the total amount of the order, including taxes and fees
- discount_amount: a decimal indicating the amount of discount applied to the order, if any
- payment_mode: a string indicating the mode of payment for the order, such as “cash”, “card”, “wallet”, etc.

- rating: an integer indicating the rating given by the customer for the order, from 1 to 5

Dimension table: Customers

- customer_id (PK): a unique identifier for each customer
- name: a string indicating the name of the customer
- phone: a string indicating the phone number of the customer
- email: a string indicating the email address of the customer
- address: a string indicating the address of the customer
- city: a string indicating the city where the customer is located
- state: a string indicating the state where the customer is located
- country: a string indicating the country where the customer is located

Dimension table: Restaurants

- restaurant_id (PK): a unique identifier for each restaurant
- name: a string indicating the name of the restaurant
- phone: a string indicating the phone number of the restaurant
- email: a string indicating the email address of the restaurant
- address: a string indicating the address of the restaurant
- city: a string indicating the city where the restaurant is located
- state: a string indicating the state where the restaurant is located
- country: a string indicating the country where the restaurant is located
- cuisine: a string indicating the type of cuisine offered by the restaurant, such as “Indian”, “Chinese”, “Italian”, etc.
- rating: an integer indicating the average rating given by customers for the restaurant, from 1 to 5

Dimension table: Delivery_Persons

- delivery_person_id (PK): a unique identifier for each delivery person
- name: a string indicating the name of the delivery person
- phone: a string indicating the phone number of the delivery person
- email: a string indicating the email address of the delivery person

3. Design a Data Warehouse for cab ride service like Uber, Lyft **(Asked in Google for Data Engineer role)**

Following is a data warehouse for a cab ride service like Uber or Lyft. Here is a possible schema:

Fact table: Rides

- ride_id (PK): a unique identifier for each ride
- customer_id (FK): a foreign key referencing the Customers dimension table
- driver_id (FK): a foreign key referencing the Drivers dimension table
- vehicle_id (FK): a foreign key referencing the Vehicles dimension table
- start_date: a date indicating the date of the ride
- start_time: a time indicating the time of the ride
- end_date: a date indicating the date of the end of the ride
- end_time: a time indicating the time of the end of the ride
- start_location: a string indicating the address or landmark of the start location of the ride
- end_location: a string indicating the address or landmark of the end location of the ride
- distance: a decimal indicating the distance covered by the ride, in kilometers or miles
- duration: an integer indicating the duration of the ride, in minutes
- fare: a decimal indicating the fare charged for the ride, including taxes and fees
- tip: a decimal indicating the tip given by the customer to the driver, if any
- payment_mode: a string indicating the mode of payment for the ride, such as “cash”, “card”, “wallet”, etc.
- rating: an integer indicating the rating given by the customer for the ride, from 1 to 5

Dimension table: Customers

- customer_id (PK): a unique identifier for each customer
- name: a string indicating the name of the customer
- phone: a string indicating the phone number of the customer
- email: a string indicating the email address of the customer

Dimension table: Drivers

- driver_id (PK): a unique identifier for each driver
- name: a string indicating the name of the driver
- phone: a string indicating the phone number of the driver
- email: a string indicating the email address of the driver

4. Design a Data Warehouse for Restaurent table booking app like Dineout **(Asked in McKinsey for Consultant Data Engineer role)**

Following is a data warehouse for a restaurant table booking app like Dineout. Here is a possible schema:

Fact table: Bookings

- booking_id (PK): a unique identifier for each booking
- customer_id (FK): a foreign key referencing the Customers dimension table
- restaurant_id (FK): a foreign key referencing the Restaurants dimension table
- table_id (FK): a foreign key referencing the Tables dimension table
- booking_date: a date indicating the date of the booking
- booking_time: a time indicating the time of the booking
- status: a string indicating the status of the booking, such as “confirmed”, “cancelled”, “no-show”, etc.
- number_of_people: an integer indicating the number of people in the booking
- total_amount: a decimal indicating the total amount of the booking, including taxes and fees
- discount_amount: a decimal indicating the amount of discount applied to the booking, if any
- payment_mode: a string indicating the mode of payment for the booking, such as “cash”, “card”, “wallet”, etc.
- rating: an integer indicating the rating given by the customer for the booking, from 1 to 5

Dimension table: Customers

- customer_id (PK): a unique identifier for each customer
- name: a string indicating the name of the customer
- phone: a string indicating the phone number of the customer
- email: a string indicating the email address of the customer

Dimension table: Restaurants

- restaurant_id (PK): a unique identifier for each restaurant
- name: a string indicating the name of the restaurant
- phone: a string indicating the phone number of the restaurant
- email: a string indicating the email address of the restaurant
- address: a string indicating the address of the restaurant
- city: a string indicating the city where the restaurant is located
- state: a string indicating the state where the restaurant is located
- country: a string indicating the country where the restaurant is located
- cuisine: a string indicating the type of cuisine offered by the restaurant, such as “Indian”, “Chinese”, “Italian”, etc.

- rating: an integer indicating the average rating given by customers for the restaurant, from 1 to 5

Dimension table: Tables

- table_id (PK): a unique identifier for each table

5. Design a Data Warehouse for Covid Vaccination Application **(Asked in Livsapce for Data Engineer role)**

Following is a data warehouse for a Covid vaccination application. Here is a possible schema:

Fact table: Vaccinations

- vaccination_id (PK): a unique identifier for each vaccination
- person_id (FK): a foreign key referencing the Persons dimension table
- vaccine_id (FK): a foreign key referencing the Vaccines dimension table
- center_id (FK): a foreign key referencing the Centers dimension table
- vaccination_date: a date indicating the date of the vaccination
- vaccination_time: a time indicating the time of the vaccination
- dose_number: an integer indicating the dose number of the vaccination, such as 1 or 2
- adverse_reaction: a string indicating the adverse reaction reported by the person after the vaccination, if any

Dimension table: Persons

- person_id (PK): a unique identifier for each person
- name: a string indicating the name of the person
- phone: a string indicating the phone number of the person
- email: a string indicating the email address of the person
- age: an integer indicating the age of the person
- gender: a string indicating the gender of the person
- address: a string indicating the address of the person
- city: a string indicating the city where the person is located
- state: a string indicating the state where the person is located
- country: a string indicating the country where the person is located

Dimension table: Vaccines

- vaccine_id (PK): a unique identifier for each vaccine
- name: a string indicating the name of the vaccine, such as “Covishield”, “Covaxin”, “Pfizer”, etc.

- manufacturer: a string indicating the name of the manufacturer of the vaccine, such as “Serum Institute of India”, “Bharat Biotech”, “Pfizer Inc.”, etc.
- efficacy: a decimal indicating the efficacy rate of the vaccine, such as 0.81, 0.78, 0.95, etc.

Dimension table: Centers

- center_id (PK): a unique identifier for each center
- name: a string indicating the name of the center, such as “Apollo Hospital”, “Max Hospital”, “Government Hospital”, etc.
- phone: a string indicating the phone number of the center
- email: a string indicating the email address of the center
- address: a string indicating the address of the center
- city: a string indicating the city where the center is located
- state: a string indicating the state where the center is located
- country: a string indicating the country where the center is located