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)

Que1. Design a Data Warehouse for IPL Cricket Tournament

Ans-1-To design a data warehouse for the IPL Cricket Tournament, we need to identify the key entities, their attributes. and the relationships between them. Based on the requirements. we can design the fact and dimension tables, establish primary and foreign keys, and follow a star schema design

VenueDim: This table stores information about the venues where the matches are played. Attributes:

venue id (Primary Key): Unique identifier for each venue.

venue name: Name of the venue.

city: City where the venue is located.

state: State where the venue is located.

country: Country where the venue is located.

TeamDim: This table stores information about the teams participating in the IPL tournament. Attributes:

team id (Primary Key): Unique identifier for each team.

team name: Name of the team. city: City where the team is based.

state: State where the team is based.

country: Country where the team is based.

Fact Table:

MatchFact: This table represents individual matches played in the IPL tournament and contains metrics related to each match. Attributes:

match id (Primary Key): Unique identifier for each match.

date: Date of the match.

venue id (Foreign Key): Identifier for the match venue.

team1 id (Foreign Key): Identifier for the first team participating in the match.

team2 id (Foreign Key): Identifier for the second team participating in the match. winner id (Foreign Key): Identifier for the winning team.

player of the match id (Foreign Key): Identifier for the player of the match.

runs scored: Total runs scored in the match. wickets taken: Total wickets taken in the match.

extras: Total extra runs in the match (like wides, no balls).

result: Result of the match (e.g., win, tie, abandoned).

PlayerDim: This table stores information about the players participating in the IPL tournament. Attributes:

player id (Primary Key): Unique identifier for each player. player name: Name of the player.

team id (Foreign Key): Identifier for the team the player belongs to.

age: Age of the playe

Total matches played by a team:

sql

Copy code

SELECT team name, COUNT(*) AS total matches

FROM MatchFact

JOIN TeamDim ON MatchFact.team1_id = TeamDim.team_id OR MatchFact.team2_id = TeamDim.team_id

GROUP BY team name;

Total runs scored by a team in a given season:

sql

Copy code

SELECT team_name, SUM(runs_scored) AS total_runs

FROM MatchFact

JOIN TeamDim ON MatchFact.team1 id = TeamDim.team id OR MatchFact.team2 id = TeamDim.team id

WHERE season = '2023'

GROUP BY team name;

Player with the most player-of-the-match awards:

SELECT player_name, COUNT(*) AS total_awards

FROM MatchFact

JOIN PlayerDim ON MatchFact.player of the match id = PlayerDim.player id

GROUP BY player name

ORDER BY total_awards DESC

LIMIT 1:

Average runs scored per match by a team in a given season:

SELECT team name, AVG(runs scored) AS average runs

FROM MatchFact

JOIN TeamDim ON MatchFact.team1 id = TeamDim.team id OR MatchFact.team2 id = TeamDim.team id

WHERE season = '2023'

GROUP BY team name;

Matches won by a team:

SELECT team name, COUNT(*) AS total wins

FROM MatchFact

JOIN TeamDim ON MatchFact.winner id = TeamDim.team id

GROUP BY team name;

These queries provide insights into various metrics, such as total matches played, total runs scored, player performance, average runs, and total wins, which can be used to analyze the IPL Cricket Tournament data.

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).

Que2.Design a Data Warehouse for Food delivery app like Swiggy, Zomato

Ans-2To design a data warehouse for a food delivery app like Swiggy or Zomato, we need to identify the key entities, their attributes, and the relationships between them. Based on the requirements, we can design the fact and dimension tables, establish primary and foreign keys, and follow a star schema design.

CustomerDim: This table stores information about the customers using the food delivery app. Attributes: customer id (Primary Key): Unique identifier for each customer. customer name: Name of the customer. contact number: Contact number of the customer. email: Email address of the customer. city: City where the customer is located. state: State where the customer is located. country: Country where the customer is located.

Fact Table: OrderFact: This table represents individual food orders placed through the app and contains metrics related to each order. Attributes: order id (Primary Key): Unique identifier for each order. customer id (Foreign Key): Identifier for the customer placing the order. restaurant id (Foreign Key): Identifier for the restaurant fulfilling the order. order date: Date and time of the order placement. delivery date: Date and time of the order delivery. order status: Current status of the order (e.g., placed, confirmed, delivered, canceled). total amount: Total amount paid for the order. payment method: Payment method used for the order.

RestaurantDim: This table stores information about the restaurants available on the food delivery app.

Attributes:

restaurant_id (Primary Key): Unique identifier for each restaurant.

restaurant_name: Name of the restaurant.

city: City where the restaurant is located.

state: State where the restaurant is located.

country: Country where the restaurant is located

ProductDim: This table stores information about the food products available for order.

Attributes:

product_id (Primary Key): Unique identifier for each product.

product_name: Name of the product.

category: Category of the product (e.g., pizza, burger, Chinese).

price: Price of the product. restaurant id (Foreign Key):

Identifier for the restaurant offering the product

Total revenue generated by the food delivery app:

SELECT SUM(total amount) AS total revenue

FROM OrderFact:

Top-selling products by quantity:

SELECT product name, SUM(quantity) AS total quantity

FROM OrderFact

JOIN OrderItemFact ON OrderFact.order_id = OrderItemFact.order_id

JOIN ProductDim ON OrderItemFact.product id = ProductDim.product id

GROUP BY product name

ORDER BY total quantity DESC

LIMIT 5:

Total orders placed by a customer:

SELECT customer name, COUNT(*) AS total orders

FROM OrderFact

JOIN CustomerDim ON OrderFact.customer id = CustomerDim.customer id

GROUP BY customer name;

Average order value by city:

SELECT city, AVG(total amount) AS average order value

FROM OrderFact

JOIN CustomerDim ON OrderFact.customer id = CustomerDim.customer id

GROUP BY city;

These queries provide insights into various metrics, such as total revenue, top-selling products, customer behavior, and average order value, which can be used to analyze the performance of the food delivery app.

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).

Que3-Design a Data Warehouse for cab ride service like Uber, Lyft

Ans3-To design a data warehouse for a cab ride service like Uber or Lyft, we need to identify the key entities, their attributes, and the relationships between them.

Based on the requirements, we can design the fact and dimension tables, establish primary and foreign keys, and follow a star schema design

PassengerDim: This table stores information about the passengers using the cab ride service. Attributes: passenger_id (Primary Key): Unique identifier for each passenger. passenger_name: Name of the passenger. contact_number: Contact number of the passenger. email: Email address of the passenger. city: City where the passenger is located. state: State where the passenger is located. country: Country where the passenger is located.

driver_id (Primary Key): Unique identifier for each driver.
driver_name: Name of the driver.
contact_number: Contact number of the driver.
email: Email address of the driver.
city: City where the driver is located.
state: State where the driver is located.

country: Country where the driver is located.

Fact Table:

RideFact: This table represents individual rides taken by passengers and contains metrics related to each ride.

Attributes:

ride_id (Primary Key): Unique identifier for each ride.

passenger_id (Foreign Key): Identifier for the passenger taking the ride.

driver_id (Foreign Key): Identifier for the driver providing the ride.

start_datetime: Date and time of the ride start.

end_datetime: Date and time of the ride completion.

distance: Distance traveled during the ride. fare_amount: Fare charged for the ride. payment_method: Payment method used for the ride.

ride_rating: Rating given by the passenger for the ride experience.

VehicleDim: This table stores information about the vehicles used for cab rides.

Attributes:

vehicle_id (Primary Key): Unique identifier for each vehicle.

vehicle_model: Model of the vehicle.

vehicle_type: Type of the vehicle (e.g., sedan, SUV). vehicle number: Registration

number of the vehicle. driver_id (Foreign Key): Identifier for the driver

associated with the vehicle.

Total rides completed:

SELECT COUNT(*) AS total_rides

FROM RideFact:

Average fare amount per ride:

SELECT AVG(fare amount) AS average fare amount

FROM RideFact:

Top-rated drivers by average rating:

SELECT driver name, AVG(ride rating) AS average rating

FROM RideFact

JOIN DriverDim ON RideFact.driver id = DriverDim.driver id

GROUP BY driver name

ORDER BY average_rating DESC

LIMIT 5;

Total rides taken by a passenger:

SELECT passenger name, COUNT(*) AS total rides

FROM RideFact

JOIN PassengerDim ON RideFact.passenger_id = PassengerDim.passenger_id

GROUP BY passenger name;

These queries provide insights into various metrics, such as total rides, average fare amount, driver performance, and passenger behavior, which can be used to analyze the cab ride service and make data-driven decisions.

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)

 Que4- Design a Data Warehouse for Restaurant table booking app like Dine out

Ans4-To design a data warehouse for a restaurant table booking app like Dine out, we need to identify the key entities, their attributes, and the relationships between them. Based on the requirements, we can design the fact and dimension tables, establish primary and foreign keys, and follow a star schema design.

CustomerDim: This table stores information about the customers using the table booking app. Attributes: customer id (Primary Key): Unique identifier for each customer. customer name: Name of the customer. contact number: Contact number of the customer. email: Email address of the customer. city: City where the customer is located. state: State where the customer is located. country: Country where the customer is located.

Fact Table:

BookingFact: This table represents individual table bookings made through the app and contains metrics related to each booking.

Attributes:

booking_id (Primary Key): Unique identifier for each booking.
customer_id (Foreign Key): Identifier for the customer making the booking.
restaurant_id (Foreign Key): Identifier for the restaurant where the booking is made.

booking_date: Date of the booking. booking_time: Time of the booking. party_size: Number of people in the booking party.

status: Current status of the booking (e.g., confirmed, canceled)

RestaurantDim: This table stores information about the restaurants available on the table booking app.

Attributes:

restaurant_id (Primary Key): Unique identifier for each restaurant.

restaurant_name: Name of the restaurant.

city: City where the restaurant is located.

state: State where the restaurant is located. country: Country where the restaurant is located

Total bookings made:

SELECT COUNT(*) AS total bookings

FROM BookingFact;

Popular restaurants by total bookings:

SELECT restaurant name, COUNT(*) AS total bookings

FROM BookingFact

JOIN RestaurantDim ON BookingFact.restaurant id = RestaurantDim.restaurant id

GROUP BY restaurant name

ORDER BY total bookings DESC

LIMIT 5;

Total bookings by customers

SELECT customer name, COUNT(*) AS total bookings

FROM BookingFact

JOIN CustomerDim ON BookingFact.customer id = CustomerDim.customer id

GROUP BY customer name;

Average party size for bookings:

SELECT AVG(party size) AS average party size

FROM BookingFact;

These queries provide insights into various metrics, such as total bookings, popular restaurants, customer behavior, and average party size, which can be used to analyze the performance of the table booking app and improve the user experience.

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)

 Que5-Design a Data Warehouse for Covid Vaccination Application

Ans-5 To design a data warehouse for a Covid Vaccination Application, we need to identify the key entities, their attributes, and the relationships between them.

Based on the requirements, we can design the fact and dimension tables, establish primary and foreign keys, and follow a star schema design. Here's a suggested design:

PatientDim: This table stores information about the patients receiving the Covid vaccination. Attributes: patient id (Primary Key): Unique identifier for each patient. patient name: Name of the patient. contact number: Contact number of the patient. email: Email address of the patient. age: Age of the patient. city: City where the patient is located. state: State where the patient is located. country: Country where the patient is located.

Fact Table: VaccinationFact: This table represents individual vaccination events and contains metrics related to each vaccination. Attributes: vaccination id (Primary Key): Unique identifier for each vaccination event. patient id (Foreign Key): Identifier for the patient receiving the vaccination. healthcare provider id (Foreign Key): Identifier for the healthcare provider administering the vaccination. vaccination date: Date of the vaccination. vaccination time: Time of the vaccination. vaccine type: Type of vaccine administered. dose number: Indicates the dose number (e.g., first dose, second dose). vaccination status: Current status of the vaccination (e.g., completed, canceled)

HealthcareProviderDim: This table stores information about the healthcare providers administering the Covid vaccinations. Attributes: healthcare provider id (Primary Key): Unique identifier for each healthcare provider. healthcare provider name: Name of the healthcare provider. contact number: Contact number of the healthcare provider. email: Email address of the healthcare provider. facility name: Name of the healthcare facility. city: City where the healthcare facility is located. state: State where the healthcare facility is located. country: Country where the healthcare facility is located.

Total vaccinations administered:

SELECT COUNT(*) AS total vaccinations

FROM VaccinationFact:

Popular vaccine types by total vaccinations:

SELECT vaccine type, COUNT(*) AS total vaccinations

FROM VaccinationFact

GROUP BY vaccine type

ORDER BY total_vaccinations DESC

LIMIT 5;

Total vaccinations by patient:

SELECT patient name, COUNT(*) AS total vaccinations

FROM VaccinationFact

JOIN PatientDim ON VaccinationFact.patient id = PatientDim.patient id

GROUP BY patient name;

Vaccination completion rate by healthcare provider:

SELECT healthcare provider name,

COUNT(CASE WHEN vaccination status = 'completed' THEN 1 END) AS completed vaccinations,

COUNT(*) AS total vaccinations

FROM VaccinationFact

JOIN HealthcareProviderDim ON VaccinationFact.healthcare_provider_id = HealthcareProviderDim.healthcare_provider_id GROUP BY healthcare provider name;

These queries provide insights into various metrics, such as total vaccinations, popular vaccine types, patient vaccination statistics, and completion rates by healthcare providers, which can be used to monitor the progress and effectiveness of the Covid vaccination program.