Design a Data Warehouse for Restaurent table booking 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)

a) Design Fact & Dimension tables

The relevant entities for each dimension table and the fact table are:

Dimension Table: User
- User ID
- Name
- Email
- Phone Number
- Loyalty Status

Dimension Table: Date
- Date ID
- Day of the Week
- Month
- Year

Dimension Table: Booking Revenue
- Booking ID
- Total Amount
- Discounts Applied
- Payment Method

Dimension Table: Restaurant
- Restaurant ID
- Name
- Location
- Cuisine Type
- Average Rating

Dimension Table: Time

- Time ID
- Start Time
- End Time

Dimension Table: User Review

- Review ID
- User ID
- Restaurant ID
- Review Text
- Rating
- Timestamp

Fact Table: Table Booking

- Booking ID
- User ID
- Restaurant ID
- Date ID
- Time ID
- Review ID

These entities represent the key attributes for each dimension table and the fact table, providing the necessary information for analysis and reporting in a simplified manner.

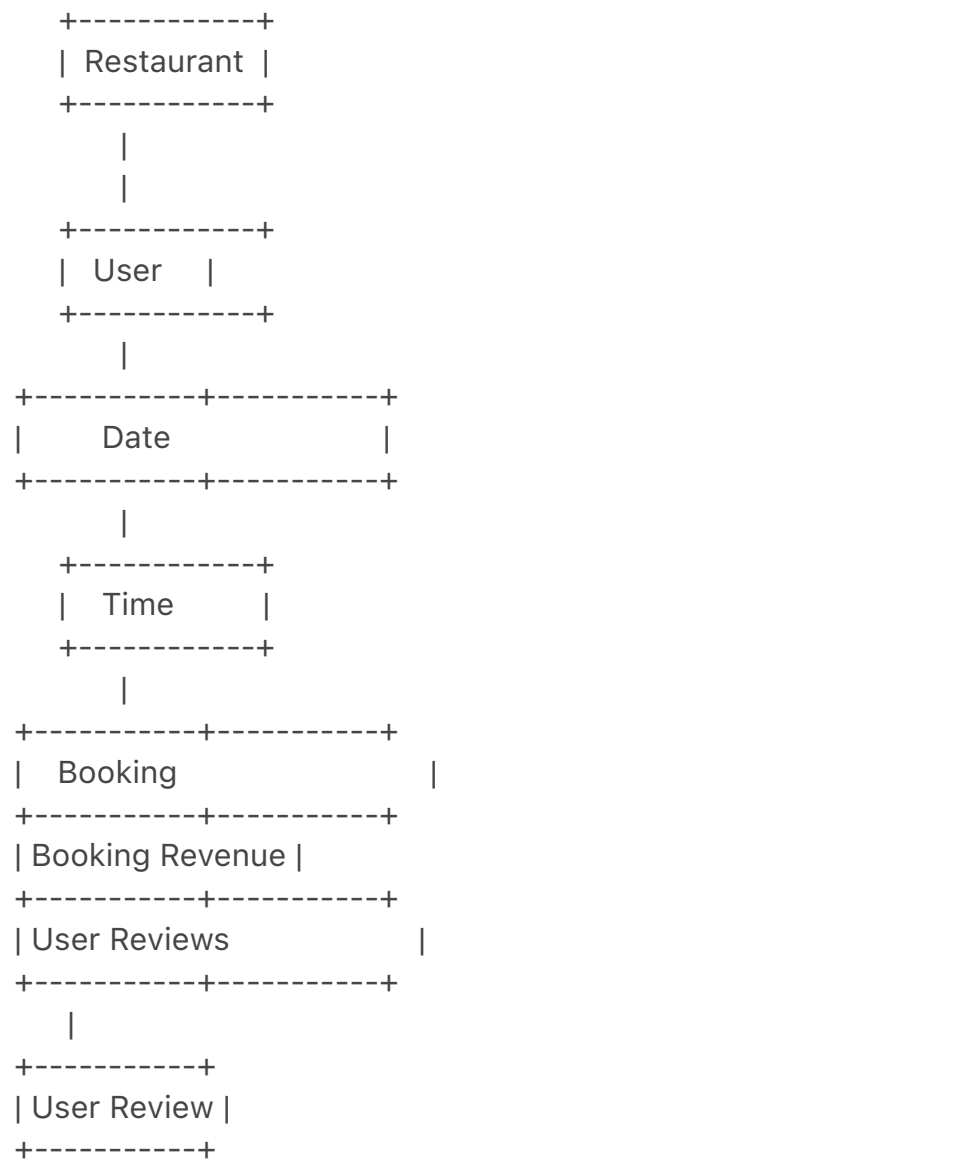
b) Create meaningful Primary & Foreign keys

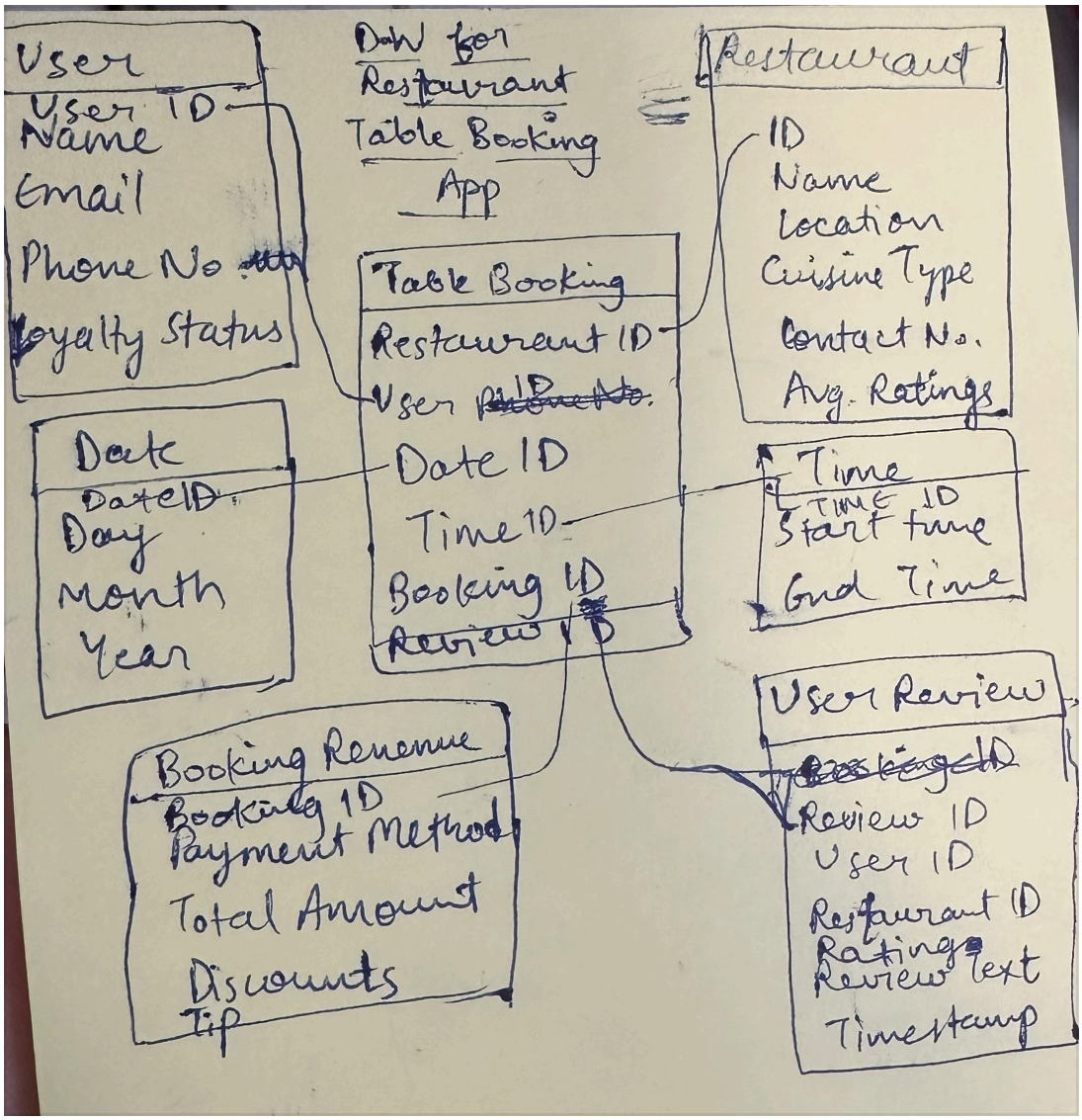
```
-- Dimension Table: User
CREATE TABLE User (
  UserID INT PRIMARY KEY,
  Name VARCHAR(50),
  Email VARCHAR(100),
  PhoneNumber VARCHAR(20),
  LoyaltyStatus VARCHAR(50)
);

-- Dimension Table: Date
CREATE TABLE Date (
  DateID INT PRIMARY KEY,
  DayOfWeek VARCHAR(20),
  Month VARCHAR(20),
  Year INT
```

```
);

-- Dimension Table: PaymentMethod
 CREATE TABLE PaymentMethod (
 PaymentMethodID INT PRIMARY KEY
 );

-- Dimension Table: Booking Revenue
CREATE TABLE BookingRevenue (
 BookingID INT PRIMARY KEY,
 TotalAmount DECIMAL(10, 2),
 DiscountsApplied DECIMAL(10, 2),
 PaymentMethodID INT,
 FOREIGN KEY (PaymentMethodID) REFERENCES
PaymentMethod(PaymentMethodID)
);

-- Dimension Table: Restaurant
CREATE TABLE Restaurant (
 RestaurantID INT PRIMARY KEY,
 Name VARCHAR(100),
 Location VARCHAR(100),
 CuisineType VARCHAR(50),
 AverageRating DECIMAL(3, 2)
);

-- Dimension Table: Time
CREATE TABLE Time (
 TimeID INT PRIMARY KEY,
 StartTime TIME,
 EndTime TIME
);

-- Dimension Table: User Review
CREATE TABLE UserReview (
 ReviewID INT PRIMARY KEY,
 UserID INT,
 RestaurantID INT,
 ReviewText TEXT,
 Rating DECIMAL(2, 1),
 Timestamp TIMESTAMP,
 FOREIGN KEY (UserID) REFERENCES User(UserID),
 FOREIGN KEY (RestaurantID) REFERENCES Restaurant(RestaurantID)
);
```

-- Fact Table: Table Booking

```
CREATE TABLE TableBooking (
  BookingID INT PRIMARY KEY,
  UserID INT,
  RestaurantID INT,
  DateID INT,
  TimeID INT,
  ReviewID INT,
  FOREIGN KEY (UserID) REFERENCES User(UserID),
  FOREIGN KEY (RestaurantID) REFERENCES Restaurant(RestaurantID),
  FOREIGN KEY (DateID) REFERENCES Date(DateID),
  FOREIGN KEY (TimeID) REFERENCES Time(TimeID),
  FOREIGN KEY (ReviewID) REFERENCES UserReview(ReviewID)
);
```





d) SQL queries:

Write a Query in SQL to find the customers that came atleast one time every month in the past one year and did a transaction of atleast $200 per month

```
SELECT User.UserID, User.Name
FROM User
INNER JOIN TableBooking ON User.UserID = TableBooking.UserID
INNER JOIN Date ON TableBooking.DateID = Date.DateID
```

```
INNER JOIN BookingRevenue ON TableBooking.BookingID =
BookingRevenue.BookingID
WHERE Date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
GROUP BY User.UserID, User.Name
HAVING COUNT(DISTINCT EXTRACT(YEAR_MONTH FROM Date)) = 12
  AND SUM(BookingRevenue.TotalAmount) >= 200
```

It filters the data based on the past year and then groups the results by the user. The HAVING clause checks if the user has made at least one transaction every month for the past year and if the sum of their transaction amounts is at least $200 per month.

Find customers who came at least once every quarter in the past two years and made a transaction of at least $500 per quarter.

```
SELECT User.UserID, User.Name
FROM User
INNER JOIN TableBooking ON User.UserID = TableBooking.UserID
INNER JOIN Date ON TableBooking.DateID = Date.DateID
INNER JOIN BookingRevenue ON TableBooking.BookingID =
BookingRevenue.BookingID
WHERE Date >= DATE_SUB(CURDATE(), INTERVAL 2 YEAR)
GROUP BY User.UserID, User.Name
HAVING COUNT(DISTINCT EXTRACT(YEAR_QUARTER FROM Date)) = 8
  AND SUM(BookingRevenue.TotalAmount) >= 500
```

Find customers who made the highest total revenue in the past year:

```
SELECT User.UserID, User.Name, SUM(BookingRevenue.TotalAmount) AS
TotalRevenue
FROM User
INNER JOIN TableBooking ON User.UserID = TableBooking.UserID
INNER JOIN BookingRevenue ON TableBooking.BookingID =
BookingRevenue.BookingID
WHERE TableBooking.DateID >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
GROUP BY User.UserID, User.Name
ORDER BY TotalRevenue DESC
LIMIT 10;
```