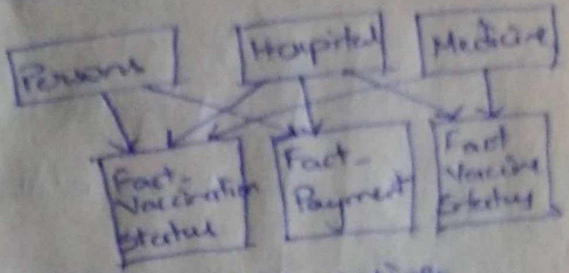
Datawarehouse for Covid Application

Dimension Tables



→ Here we are using Star Schema

~~Persons~~ Pa

Person_id
Name
Phone_Number
Adhaar_Number

Primary Key → Person_id, Adhaar_Numer

Hospital

Hospital_id
Hospital_Name
Location
Address
Hospital_Type

Primary Key → Hospital_id, Hospital_Name, Hospital_Type

Medicine

Medicine_id
Medicine_name
Manufacturer_name

↓
Primary Key
↓
Medicine_id, Medicine_Name, Manufacturer_Name

Fact Tables

Fact_Vaccination_Status

Vaccination_id
Person_id
Hospital_id
Pre_Dosage status
Medicine_id
Post_Dosage_status

↓
Primary Key : Vaccination_id
Foreign_Key : Person_id, Hospital_id, Medicine_id

Fact_Payment

Payment_id
Person_id
Hospital_id
Vaccination_id
Payment_status
Amount_paid

Primary_Key : Payment_id, Payment_status
Foreign_Key : Person_id, Hospital_id, Vaccination_id

Fact_Vaccine_Status

Hospital_id

Medisine_id

Number_of_vaccines

primary Key :- Number_of_vaccines

Foreign Key :- Hospital_id, Medicine_id

SQL Queries

1) In order to check the most_visited Hospital

Select Hospital_id, count(*) from Fact_Vaccination_statu
group by Hospital_id order by 2 desc.

2) To find number of vaccines in each hospital

Select * from Fact_Vaccine_status

3) To find the person with the vaccination_status after visit

Select person_id from Fact_vaccination_status
where post_vaccination_status = 0/1/2/3

4) To find the most used vaccine

Select medicine_id, count(*) from Fact_vaccinatio
status groupby 1 order by 2, desc.

5) To check the revenue per hospital.

Select hospital_id, count(*) from Fact_payment
where payment_status = "Y"
group by 1
order by 2 desc

Datawarehouse for Restaurant Table Booking

Customer

Customer_id
Name
Phone_Number
Email

Primary-Key :- Customerid

location

location_id
Area
City
Address
Landmark

↓
Primary-Key
↓
location_id, Address

Restaurent

Restaurent_id
Restaurent_Name
Restaurent_Type
Price-per-hour
location_id

↓
Primary-Key : Restaurent_id, Restaurent_Name
Price_per_hour

→ Price-per-hour might also include Taxes

Tables

Restourent_id
Table_id
Number_of_Tables

↓
Primary Key : Table_id, Number of Tables

DataWarehouse For Restaurent Table Booking

Customer

Customer_id
Name
Phone - Number
Email

Primary-Key :- Customerid

location

location_id
Area
City
Address
Landmark

↓
Primary-Key
↓
location_id, Address

Restaurent

Restaurent_id
Restaurent_Name
Restaurent_Type
Price-per-hour
location_id

↓
Primary-Key : Restaurent_id, Restaurent_Name, Price_per_hour

→ Price-per-hour might also include Taxes

Tables

Restourent_id
Table-id
Number-of-Tables

↓
Primary Key : Table_id, Number of Tables

Fact_Booking

Booking_id
Customer_id
Restaurent_id
Table_id,
location_id
Number_of_visitors
valid_from_time
valid_to_time
Booking_date

Primary_Keys :- Booking_id, Number_of_visitors, Booking_date

Foreign_Keys :- Customer_id, Restaurent_id, Table_id, location_id,

Fact_Payment

Payment_id
Booking_id
~~Customer~~
Price_per_hour
Final_price
Payment_status

Primary Keys :- Payment_id, Final_price, payment status

Foreign Key :- Booking_id ; Price_per_hour

Fact_Review

Review_id
Customer_id
Restaurent_id
locality_id
Rating

Primary Keys → Review_id, Rating
Foreign_Key → Customer_id, Restaurent_id

SQL Queries

1) To get highest rated ~~table~~ Restaurent

select ~~Hotel_id~~, Restaurent_id, max(Rating) from
Fact_Review group by 1

2) To get the highest / least booked Restaurant

select Restaurent_id, count(*) from Fact_Booking
group by 1, order by 1 limit 1
↓
For least

select Restaurent_id, count(*) from Fact_Booking
group by 1, order by 1 desc limit 1
↓
For highly rated

3) To get restaurent ~~per~~ locality based on
rating

select Restaurent_id, dense_rank()
over (partition by locality_id order by Rating)
as Rating_per_Area
from Fact_Review

~~Fact-Driver-location~~ Cab Service

DataWarehouse for

~~Driver~~-id

Customers

- customer-id
- customer_Name
- cust_phone_Number
- cust_location_id

Primary Key :- Customer_id, customer_Name, cust-phone_Number
Foreign Key :- cust_location_id

Drivers

- Driver-id
- Driver-Name
- Driver-phone_Number

Primary Keys :- Driver-id, Driver-Name, Driver-phone_N

Cust_Location

- Customer-id
- cust_location_id
- Source-Address
- Destination-Address
- source-locality

Primary Key :- cust_location_id, Source-Address, Destination-Address
Foreign_Key :- customer_id

Destination_Address

- Destination_id
- Customer-id
- Destination-Address

Primary Key :- Destionation_id, Destionation_Addre
Foreign Key :- Customer_id

Car

- Car-id
- Care Name
- Car-Type
- Car-Number
- Document-id

Primary Key
Car-id, Car-Number,
Car-Type
Foreign Key
Document-id

Driver-Location

- Driver-id
- Driver-location-id
- Driver-current-Address
- Driver-locality

Primary Key :- Driver-location-id, Driver-current_Address

Foreign Key :- Driver-id

Document

- Document-id
- Driver-id
- license_valid_from
- license_valid_to
- license_status
- License_Number

Foreign Key
Driver_id

Primary Key :- Document-id, License_Number, License_val
License_valid_to

Fact Trip

Trip_id
Customer_id
cust_location_id
Driver_id
Destination_id
Total_Distance
Trip_start_time
Trip_end_time
Trip_status

Primary Key :- Trip_id, Trip_status,
Foreign Key :- customer_id, Driver_id,
cust_location_id, Destination_id

Fact-Payment

Payment_id
Trip_id
Payment_mode
Tax
Base_Fare
Coupoun_id
Final_price
payment_status

Primary Key : Payment_id
Foreign_key : Trip_id,
Coupoun_id

Fact-Cancellation

Trip_id
cust_side
Driver_side
Cancel_id
cust_side_charges
Status

Primary Key :- Cancel_id, status
Foreign Key :- Trip_id

Fact-Rating

Trip_id
Rating_status
Rating
Customer_id

Primary Key :- Rating status
Foreign Key
$\downarrow$
Trip_id, customer_id

Coupouns

Coupoun_id
Coupoun_name
dicount

Primary Key
$\downarrow$
coupoun_id, Dicount

SQL Queries

1) In order to check the ratings

```
Select T. Driver_id , R. Ratings

from Fact_Rating R
inner join Fact - Trips T
on R. Trip_id = T. Trip_id
where Rating_status = "Y"
group by 1   order by 2 desc
```

2) In order to check to location from which bookings are maximum

```
select cust_location_id , count(*)
from Fact_Trip T
group by 1   order by 2 desc
```

3) In order to check who had highest number of bookings

```
select Driver_id, count(*)
from Fact_Trip
group by 1   order by 2 desc
```

2) In order to check the customers who had not paid cancellation charges after booking

```
Select ~~Driver_id~~ T. Customer_id , ~~count(~~

from Fact_Cancellation D
inner join Fact_Trips T
on D. Trip_id = ~~T. Tri~~ Fact_Trip
```

Data Warehouse For Food Delivery App

Customers

Cust_id
Cust_name
Cust_phone_number
Cust_location_id
Cust_Address
~~(cust_pincode)~~

rimary Key :- ~~Cust_id~~, cust_id
cust_phone_number, cust_location_id,

Restaurant -

Restaurent_id
Restaurant_Name
Restaurant_Type
Restaurant_Address

Primary Key :- Restaurent_Name
Restaurant_Type, Restaurant_Address, Restaurent_id

Delivery_Person

Person_id
Name
Languages_spoken
Phone_Number
~~(Vehicles Number)~~

Primary Key :-
Person_id, Name,
Phone_Number

Items

Item_id
Item_Name
Price
Restaurant_id
Item_rating

Primary Key :- Item_id,
Item_Name, Price
Foreign Key :- Restaurant_id

Coupoun

Coupoun_id
Coupoun_Name
Description
Discount
~~used~~

Primary Key :- coupoun_id, coupoun_Name,
Discount

Membership

Membership_id
Cust_id
Membership_Status
valid_from_date
valid_to_date

↓

Primary Key
↓
Membership_id_
Status
Foreign Key
↓
Cust_id

Fact_order

Order_id
User_id
Restaurant_id
item_id
Person_id
order_status

Primary Key :- order_id, order_status

Fact_Payment.

Payment_id
order_id
Price
Delivery_charge
Tax
Membership_status
Total_discount
Payment_status
Coupoun_status
~~Pay~~
Coupoun_id
Final_price

Primary key :- Payment_id, Membership_status, Payment_coupoun_status
Foreign_key :- Order_id, coupoun_id

Fact_Rating

Rating_id
User_id
Person_id
Rating
Item_Rating

Primary key
↓
Rating_id, Rating
Foreig key
↓
User_id, Person_id, Item_Rating

SQL <u>Queries</u>

1) In order to check the rating for who had been highest / lowest rated

```
Selectd.person_id, d.name, f. Rating
 from Fact_Rating f
inner join Delivery_Person d
on f. Person_id = d.Person_id
group by 1, 2
order by 3 desc
```

Analysis on the most used coupoun

```
Select coupoun_id , count(*)

from Fact_Payment

group by 1 order by 1
```

Checking the data for order with respect to Restaurant ~~locality~~.

```
(Select Resta)

Select ~~f~~. Restaurant_id, r. Restaurant_Name,
      count (*)
from Restaurant r
inner join
 Fact_order_f
on
 f. Restaurant_id = r. Restaurant_id
group by 1, 2
order by 3 desc
```

To find the number of highest rated items in each restaurant

```
Select Item_Name, Item_id,
dense_rank() over( partition by Restaurant_id
                order by Item_Rating) as

Food_Rating
from Items
```

Players

Player_id
Player-Name.
Player-Type
Player-Age

Primary keys → player-id, Player-Name, Player-Type, ~~Player-Age~~

Team

Team-id
Player-id
Team-Name
Team-Owner

↓
Primary-key
↓
Team-id,
Team-Name
Foreign-key
↓
player-id

Fact_Player_Score

Player-id
Number of Runs (Runs)
Number of wickets (wickets)
Boundaries
Avg_Score
Run_Rate

Primary-Keys
↓
Runs, Wickets
Foreign-Keys
↓
Player-id

Stadium

Stadium-id
Area
Stadium-Name
State
Seating-Capacity

Primary_Key: Stadium-id, stadium-Name, Area

Visitors

Visitor_id
Name
phone_Number
Email

Primary-key: Visitor-id, ~~Name~~ phone_Number

Fact_Match

Foreign-key
~~[illegible]~~ Stadium-id
↑

Match-id
Stadium-id
Team-id ~~[illegible]~~
Match_date
Match_start_time
Match-end_time
Match-status
Team_score

Primary-key
↓
Match-date, Match-start_time
status, Match-id

Fact_Visitors

Visitor_id
Stadium-id
Visitor-payment_status
Ticket_status
Ticket_type

Primary-key: visitor_payment-status, Ticket-type, Ticket-type

Foreign-Key: visitor_id, stadium-id

Fact _ Final _ Points

Team _id
Total _ points
Net _ Run _ Rate
Number _of _Wins
Number _of _ looses
Number _ of _ ties

Primary key
↓
Total _points,
Net _Run _Rate

Foreign _ key
↓
Team _id

SQL Quries

i) Inorder to check the teams that are getting qualified for quater _ final

```
Select t.Team_. Name
from Team t
inner join Fact_Final_Points f
on t.team_id = f.team_id.
~~(where f. total_points)~~
order by f. total _points desc limit 4
```

Inorder to check the stadium in which visitors are more

```
Select S. Stadium_Name, M. stadium_id,
            count(*)
from Stadium S
inner join Fact_Match M
on M. Stadium_id = S. Stadium_id
group by 1,2
order by 2 desc
```

3) Inorder to check the player with highest runs

```
select P. Player_Name

from Player P
inner join Fact_Player_Score F
on P. Player_id = F. Player_id
order by Number_of_runs desc limit 1
```

4) In order to check the player with highest number of wickets

```
select P. Player_Name

from player P
inner join Fact_Player_Score P
on P. Player_id = F. Player_id
orderby Number_of_wickets desc limit 1
```