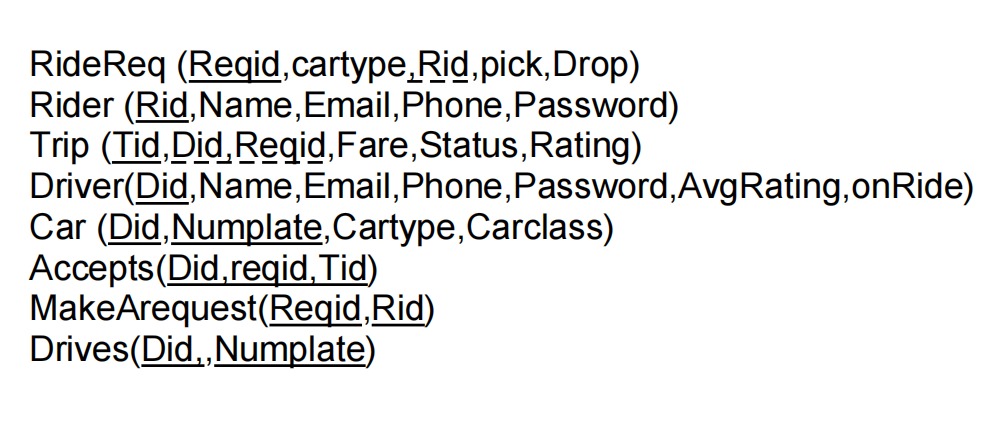
**Updated Schema**



**Grants**

create user 'head'@'localhost' identified by '123';

grant select,delete,insert on Driver to 'head'@'localhost';

create user 'admin'@'localhost' identified by '123';

grant all on Rider to 'admin'@'localhost';

**Queries**

**1. For allotting a driver**

CREATE VIEW alloted\_driver AS

SELECT driver.Did,Name, Phone, numplate,cartype

FROM driver

INNER JOIN car

INNER JOIN ride\_req

ON driver.Did=Car.Did AND car.cartype=ride\_req.Car\_type AND onride=0 AND Reqid=1

Optimisation: used INNER JOIN instead of nested queries ; column names instead of \* ;

Indexed car table on car type and driver table on onride attribute

**2. Showing ranks of drivers according to their rating**

SELECT Name , dense\_rank() over (order by Avg\_rating desc) as s\_rank

from driver

**3. Show location suggestions when the rider starts typing**

SELECT DISTINCT(loc1)

FROM fare\_chart

WHERE loc1 like 'K%';

**4. Show fares according to car type and pickup and drop location**

CREATE VIEW car\_fare AS

SELECT DISTINCT cartype, carclass

FROM car;

SELECT cartype, carclass\*fare as Fare, loc1,loc2

FROM car\_fare

CROSS JOIN( SELECT fare,loc1,loc2

FROM fare\_chart

WHERE (loc1='Karol Bagh' AND loc2='Khan Market') OR (loc2='Karol Bagh' AND loc1='Khan Market') ) sub;

Optimisation: First took a subset of fare chart table, then performed cross join

**5. Updates average rating column of driver**

UPDATE driver

SET Avg\_rating=(SELECT driver\_rating(2))

WHERE Did=2;

**6. Driver can see all their trips**

CREATE VIEW mytrips AS

SELECT ride\_req.rid, ride\_req.Date,ride\_req.pickup,ride\_req.drop, trip.tid

FROM trip

JOIN ride\_req

ON trip.Did=2 AND ride\_req.reqid=trip.reqid

**7. Show total earnings of a driver till now**

SELECT SUM( fare), Driver.Name

FROM trip,driver

WHERE trip.Did=2 AND driver.Did=2

**8. Select driver with max rating**

SELECT MAX(Avg\_rating), Driver.Name

FROM driver

**9. Show number of trips completed by driver**

SELECT COUNT(\*) AS no\_of\_trips

FROM trip

WHERE Did=4 AND status=1;

**10.Update phone number of driver**

UPDATE driver.` SET phone=758749123 where Did=3

**PL SQL**

1. **Calculates average rating of driver**

DELIMITER $$

DROP FUNCTION IF EXISTS driver\_rating $$

create function driver\_rating (D\_ID INT)

returns float

DETERMINISTIC

begin

declare rating float;

select avg(trip.Rating) into rating

from trip

where trip.Did= D\_ID;

return rating;

end $$

DELIMITER ;

1. **Checks if a driver is available or not**

DELIMITER $$

DROP FUNCTION IF EXISTS check\_deriver $$

create function check\_deriver (did int)

returns int

DETERMINISTIC

begin

declare avbl int;

select driver.onride into avbl

from driver

where driver.DiD = did;

return avbl;

end$$

DELIMITER ;

select check\_deriver(2) as avbl;

**Indexes**

Cars: cartype

Driver: onride

Rider: Phone

Ride\_req: Rid

Trip: Reqid, Status

**Triggers**

1. **Free the driver when trip ends**

DELIMITER $$

CREATE TRIGGER free\_driver

AFTER INSERT

ON payments FOR EACH ROW

BEGIN

IF(trips.status = 1 AND trip.Rid = 2)

THEN

INSERT INTO driver(status)

VALUES(0)

WHERE trips.r\_id = 2 AND driver.Did = trip.Did;

END IF;

END$$

1. **Reserve the driver when trip is ongoing**

DELIMITER $$

CREATE TRIGGER reserve\_driver

AFTER INSERT

ON trips FOR EACH ROW

BEGIN

IF(trips.status =0)

THEN

INSERT INTO drivers(status)

VALUES(1)

WHERE trips.r\_id = (currently logged in user) AND driver.d\_id = trips.d\_id;

END IF;

END$$

1. **For making trip status complete on payment**

delimiter //

CREATE TRIGGER ride\_complete

AFTER UPDATE

ON trip.fare FOR EACH ROW

UPDATE trip(status) VALUES("Complete")

END; //