MGMT 582 – Team 5 – CityBus: Route Optimization

Team Members:

Lakshminaryana Sudheendra Sarma Mudradi (Laksh)

Geethika Mittal

Abhinao Ojha

Kanak Agrawal

Phalguni Vatsa

Introduction:

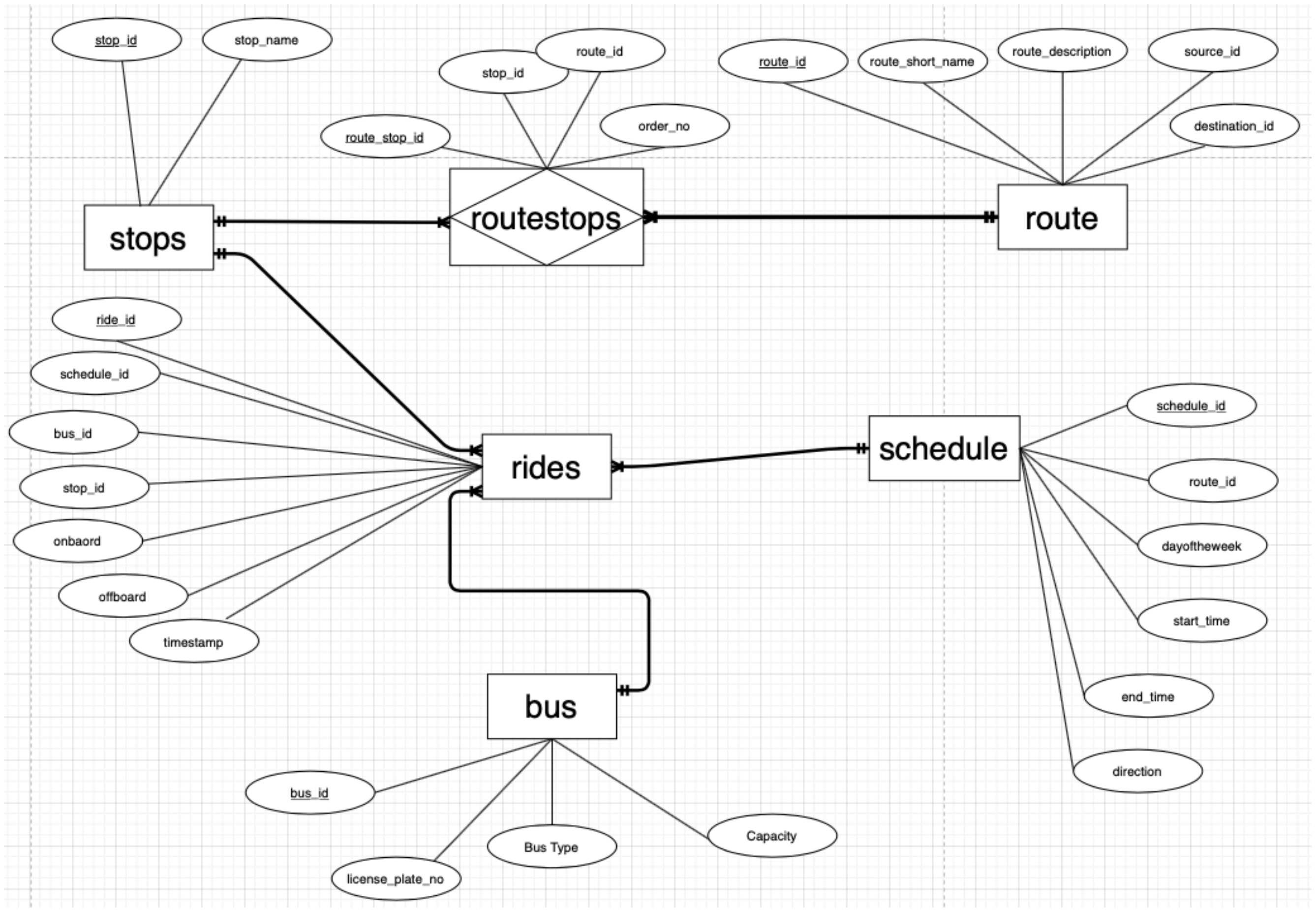
CityBus of West Lafayette, IN wants to increase its revenue by optimizing the route and resource allocation. The objective is to identify and analyse route performances which can help CityBus in creating a dynamic schedule based on the flux of passengers.
As consultants, we aim to create a managerial report summarizing the current state of operation based on the exploratory data analysis and provide insights that would help CityBus for appropriate resource allocation and route optimization.

Description of the Dataset

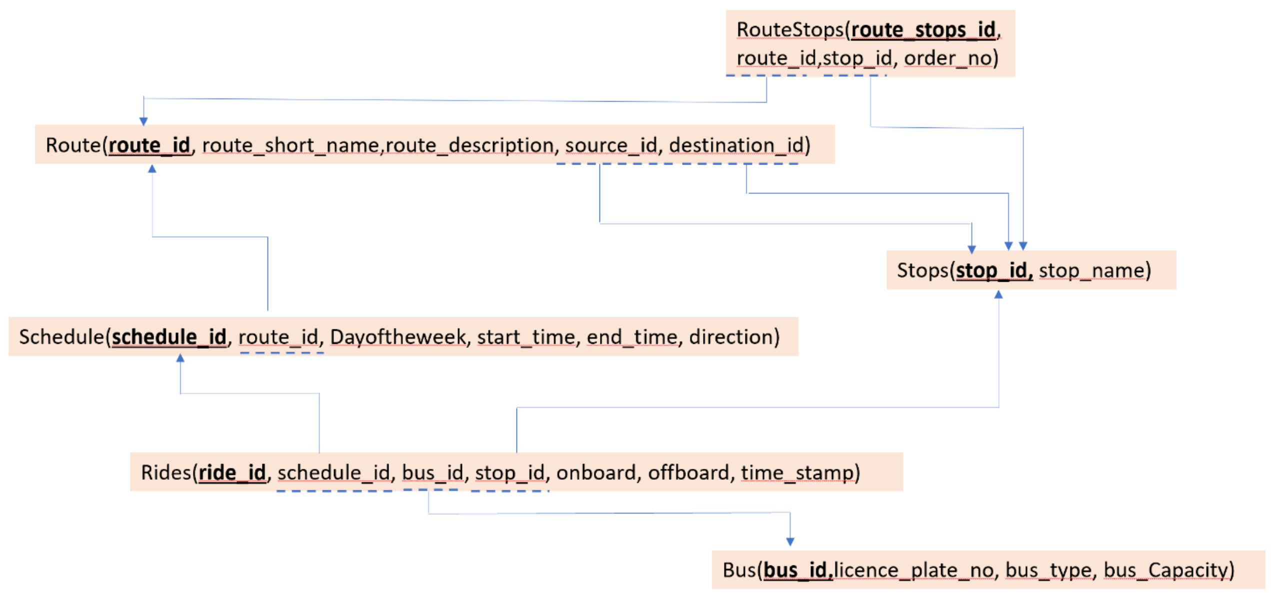
The dataset consists of the following information about the business:

1. Type and number of buses owned by CityBus.
   (a) **Type of bus** – Regular, mini-bus or double decker
   (b) **Bus Capacity** – Bus could have 20,34 or 64 passengers
   (c) **Licence plate** – Licence plate number of the physical bus
2. Routes and their corresponding stops catered to by CityBus.
   (a) **Stop Name** – List of names of all the stops with their corresponding IDs
   (b) **Route name and description** – Short name for the route and route ID along with its description.
   (c) **Route source and destination** – First and last stops of every route
   (d) **Mapping of stops to each route** - List and order of all the stops associated with a particular route
3. Bus schedule for the week
   (a) **Start and end time** for each route ID for each day of the week uniquely identified by the schedule ID
   (b) **Direction of the route** indicating the actual start and stop points of a particular ride
4. Passenger onboarding and offboarding details with timestamp for a week
   (a) **Number of passengers onboarding and deboarding** from a particular stop for each route with the timestamps
   (b) **Each entry**- unique combination of schedule ID, route ID is mapped to the physical entity bus

Entity Relationship Diagram (ERD)



Relational Schema: (Include Primary keys and secondary keys)


RouteStops(route_stops_id, route_id,stop_id, order_no)
Route(route_id, route_short_name,route_description, source_id, destination_id)
Stops(stop_id, stop_name)
Schedule(schedule_id, route_id, Dayoftheweek, start_time, end_time, direction)
Rides(ride_id, schedule_id, bus_id, stop_id, onboard, offboard, time_stamp)
Bus(bus_id,licence_plate_no, bus_type, bus_Capacity)

Relational Schema for myCityBus

Normalization analysis:

The master table for our project was of the following form:

Rides(<u>ride_id</u>, stopID, route_id, busID, scheduleID,StopName, licence_plate, bus_type, bus_capacity, Timestamp_stop, Dayoftheweek, route_short_name, route_long_name, route_desc, source, destination, order, start_time, end_time, Direction, OnBoarding, Deboarding)

Sample Table:

<u>ride_id</u>	stopID	route_id	busID	scheduleID	StopName	licence_plate	bus_type	bus_capacity	timestamp_stop	Dayoftheweek
1	BUS668	10254	BUSMZ	1	Kirby Riskk	IND342	double-dec	64	9:18	1,2,3,4,5
2	BUS648	10260	BUSMZ	2	Alabama St	IND545	mini-bus	20	11:56	1,2,3,4,5
3	BUS902	10263	BUSMZ	3	Blueberry L	IND364	regular	34	12:04	1,2,3,4,5
4	BUS387	10263	BUSMZ	4	University :	IND364	regular	34	12:10	1,2,3,4,5

route_short_name	route_desc	source	destination	order	start_time	end_time	Direction	onboardin	offboarding
1B	1B Salisbury, o	Blackbir	Northweste	7	9:00	10:00	0	4	2
7	7 South Street,	SR 26 &	1st St & Jisc	5	11:00	12:00	1	2	2
9	9 Park East - O	Blueber	Old Romne	1	12:00	13:00	1	5	0
9	9 Park East - O	Blueber	Old Romne	4	12:00	13:00	1	1	4

<u>1NF:</u> As can be observed from the above sample table, it has multi-valued attribute Dayoftheweek. This table is not in the 1NF form.
The table was thus converted to 1NF by splitting the multi-valued column into multiple rows.

2NF: The rides table is unique with ride_id as primary key as there are no partial dependencies. So, our table is already in second normal form.

3NF: Several non-key attributes were not functionally dependent on the primary key but had transitive dependency on other non-key attributes.
The details are listed below:

1. Licence_plate,bus_type and bus_capacity depended only on the busID.
2. route_short_name, route_desc, source, destination depended only on the route_id
3. start_time, end_time, Direction depended only on the scheduleID.
4. StopName was dependent only on stopID.

Thus, to convert the table to 3NF, the database was restructured to have the following tables:

Rides(ride_id,schedule_id, bus_id,stop_id, onboard,offboard, timestamp)
Route(route_id, route_short_name, route_desc, source_id, destination_id)
Bus(busID,licence_plate, bus_type, bus_capacity)
Schedule(schedule_id,route_id,DayoftheWeek,start_time,end_time,direction)
RouteStops(route_stops_id,route_id,stop_id,order_no)
Stops(stopID,StopName)

Snippets of the final tables have been added in the Appendix.

Data Model & Design:

The data model uses MySQL for structured dataset from CityBus. The dataset has relational dependencies between its entities like each bus will have routes, the routes will have its stops and each stop will have onboarding and offboarding passengers. Since each of these entities can have one to one or one to many relationships amongst each other and can be structured in a clear relationship with each other we decided to use the relational database and the SQL to perform operations on it.

The database has 5 entities namely bus, stops, schedules, routes and rides which contain all the information with a primary and foreign key in each of the tables to make querying and relations easy. The data insertion though huge but can accessed through a local SQL database server.

The relational nature of the database helps in memory optimization and querying data to understand the numbers at various level of information. MYSQL workbench being an open-source software gives all the flexibility and functionality needed by an administrator to create and save the table in the format as required.

Using relational database and SQL not only allows us to perform consecutive operations on the data it also allows us to extract insightful information which can be then used to provide the recommendations to the clients.

DDL Queries: (Creation and data insertion queries)

We aim to find the current average utilization of the bus capacity at any point, i.e., at any stop. This is achieved for a single route by selecting according to its unique schedule for the route, day of the week and time, and then calculating the number of people onboard the bus at every stop. The average people onboard the bus is then taken as a proportion of the complete bus capacity, which can be extracted by querying the bus table with the identification of the bus used for this schedule.

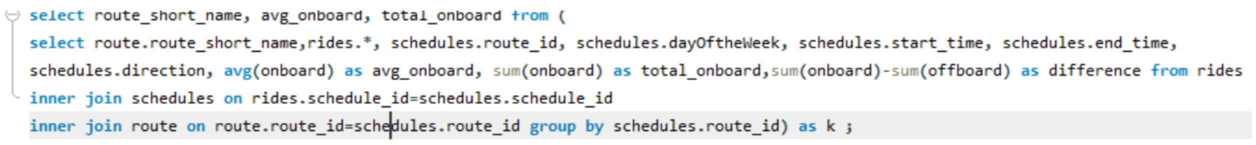
The tables that we decided to create for the problem were for the entity bus, route, rides, stops, schedule and an associative entity routestops. The creation queries for these had the primary keys, foreign keys with constraints like not null and unique. The datatypes for each column were specified with their use in mind.

For insertion queries the ids were auto-incremented through the auto-increment keyword and the rest of the columns were referred from the CityBus real life stops, buses, routes and rides. The datatype for each column was kept in check and the foreign keys were inserted through the subqueries.

Business Queries

First, we find the number of people travelling or taking the bus services in the city which is we assess the demand.

-- 1.1 Demand (People travelling) by route

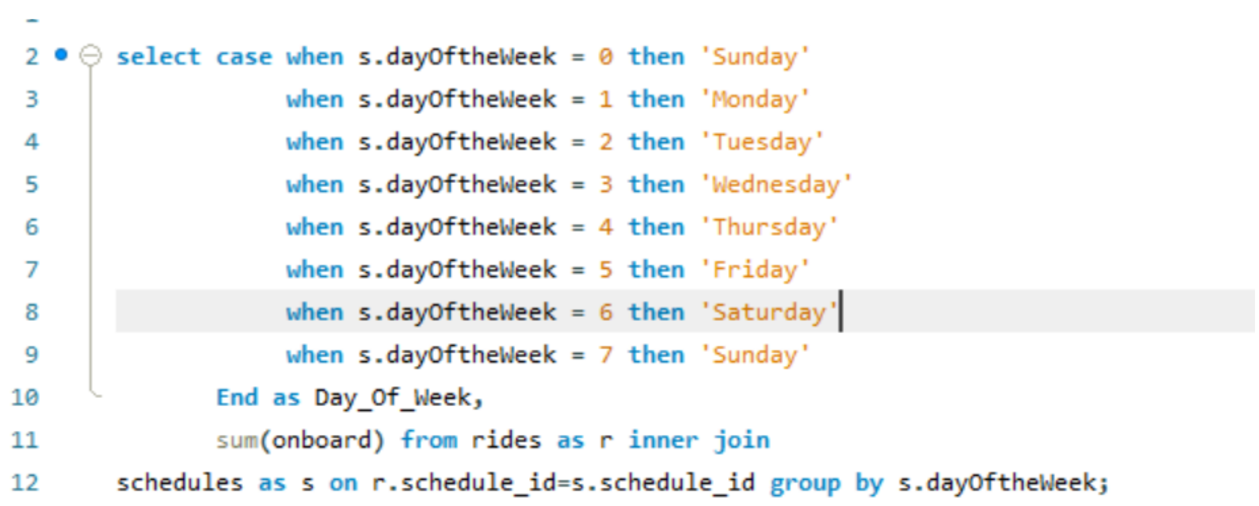


```
select route_short_name, avg_onboard, total_onboard from (
select route.route_short_name,rides.*, schedules.route_id, schedules.dayOftheWeek, schedules.start_time, schedules.end_time,
schedules.direction, avg(onboard) as avg_onboard, sum(onboard) as total_onboard,sum(onboard)-sum(offboard) as difference from rides
inner join schedules on rides.schedule_id=schedules.schedule_id
inner join route on route.route_id=schedules.route_id group by schedules.route_id) as k ;
```

Route	Avg(onboard)	Total_onboard
4A	2.0874	2411
24	3.1476	661
6A	2.0857	438
4B	3.1286	219
6B	1.8762	197
35	1.8571	195
15	2.5429	178
28	1.6286	57

With the help of this query, we could compute the demand (the number of people) using the bus at a given route. With the dynamic data captured for each ride in the rides table, we will get to know the exact number of people who onboarded or deboarded a bus at the stop, and so the total number of people that were in the bus for the route as well. So now we map this data to the schedule table to find the ride and its information so that we find with the real time data which rides on which route has been the busiest.

-- 1.2 Demand by day of week

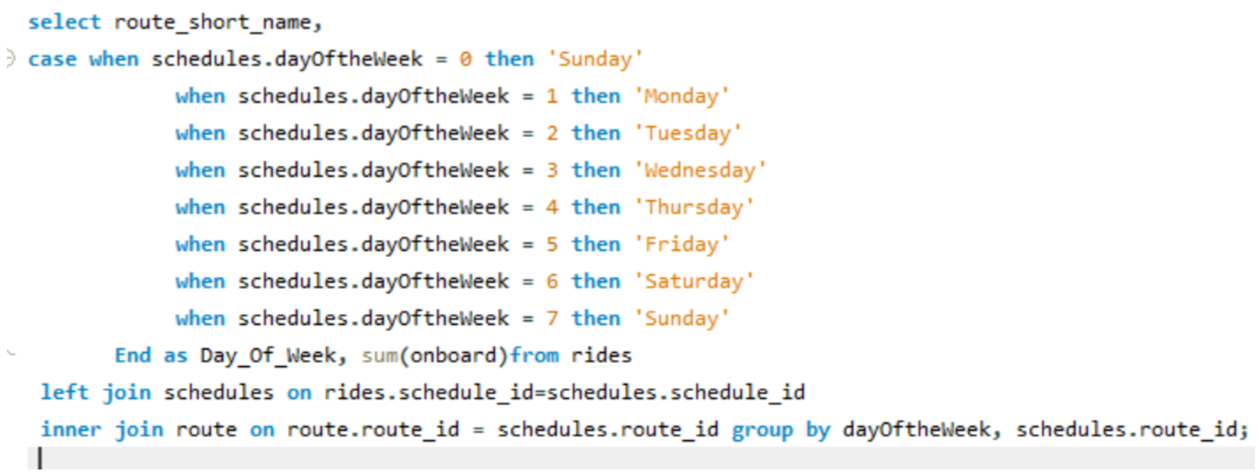


```
select case when s.dayOftheWeek = 0 then 'Sunday'
            when s.dayOftheWeek = 1 then 'Monday'
            when s.dayOftheWeek = 2 then 'Tuesday'
            when s.dayOftheWeek = 3 then 'Wednesday'
            when s.dayOftheWeek = 4 then 'Thursday'
            when s.dayOftheWeek = 5 then 'Friday'
            when s.dayOftheWeek = 6 then 'Saturday'
            when s.dayOftheWeek = 7 then 'Sunday'
        End as Day_Of_Week,
        sum(onboard) from rides as r inner join
schedules as s on r.schedule_id=s.schedule_id group by s.dayOftheWeek;
```

Day_of_Week	Sum(onboard)
Tuesday	989
Monday	936
Wednesday	701
Thursday	689
Friday	556
Saturday	275
Sunday	210

To find the number of people travelling on a particular day we have written this query where refer the real time data we capture for rides from the rides table and find the total number of people travelling on any selected day which we can get from the schedule table.

-- 1.3 Demand by route and a particular day



```
select route_short_name,
case when schedules.dayOftheWeek = 0 then 'Sunday'
            when schedules.dayOftheWeek = 1 then 'Monday'
            when schedules.dayOftheWeek = 2 then 'Tuesday'
            when schedules.dayOftheWeek = 3 then 'Wednesday'
            when schedules.dayOftheWeek = 4 then 'Thursday'
            when schedules.dayOftheWeek = 5 then 'Friday'
            when schedules.dayOftheWeek = 6 then 'Saturday'
            when schedules.dayOftheWeek = 7 then 'Sunday'
       End as Day_Of_Week, sum(onboard)from rides
 left join schedules on rides.schedule_id=schedules.schedule_id
 inner join route on route.route_id = schedules.route_id group by dayOftheWeek, schedules.route_id;
```

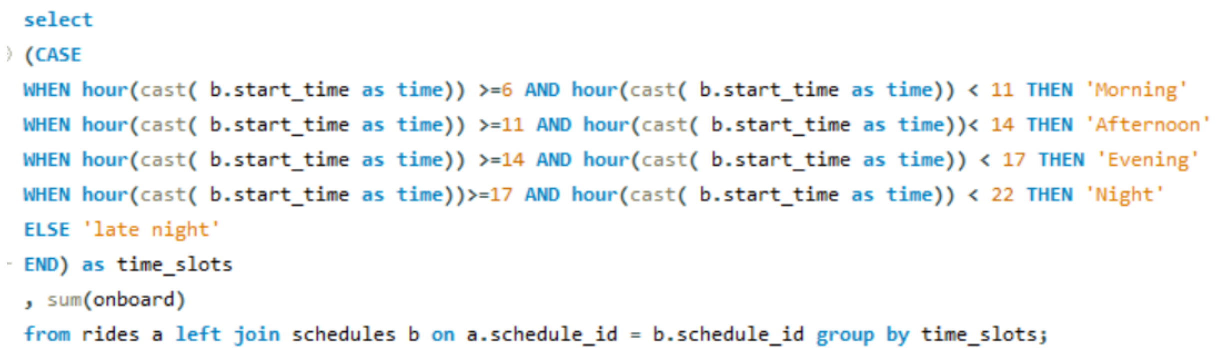
To find the total number of people travelling on a particular route on a day we have used the query where we capture the real time data of rides every day and we can then map it to the schedule table where we find the route information and the week of the day information. Joining both these tables, we can get the highest ridership.

Route_short_name	DayOfWeek	Sum(onboard)
4A	Tuesday	524
4A	Wednesday	440
4A	Thursday	377
4A	Friday	323
4A	Monday	322
4A	Saturday	275
24	Tuesday	249
15	Monday	178
24	Monday	155
4A	Sunday	150
6A	Monday	132
4B	Tuesday	132
6A	Thursday	131
24	Thursday	130
24	Wednesday	127
6A	Friday	114
4B	Monday	87
6B	Tuesday	84
35	Wednesday	73
6B	Friday	62
35	Monday	62
35	Monday	62
6A	Wednesday	61
35	Sunday	60

35	Sunday	60
28	Friday	57
28	Friday	57
6B	Thursday	51

-- 1.4 Demand by time slot

To find the number of riders using the bus services on a particular day the query uses case where we can classify the start and end times into different time slots and can then find the total number of people onboarding during those time slots. This will give us an insight into the busiest hours of the day.

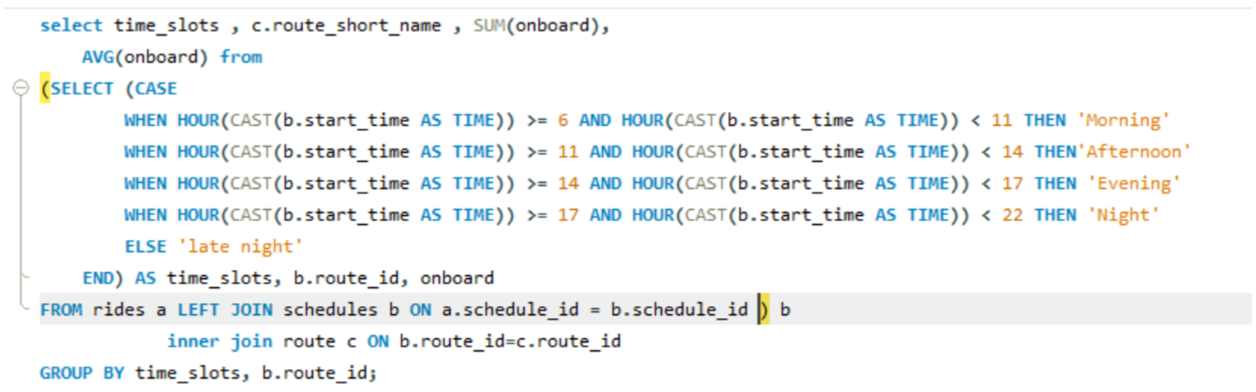


```
select
(CASE
WHEN hour(cast( b.start_time as time)) >=6 AND hour(cast( b.start_time as time)) < 11 THEN 'Morning'
WHEN hour(cast( b.start_time as time)) >=11 AND hour(cast( b.start_time as time))< 14 THEN 'Afternoon'
WHEN hour(cast( b.start_time as time)) >=14 AND hour(cast( b.start_time as time)) < 17 THEN 'Evening'
WHEN hour(cast( b.start_time as time))>=17 AND hour(cast( b.start_time as time)) < 22 THEN 'Night'
ELSE 'late night'
END) as time_slots
, sum(onboard)
from rides a left join schedules b on a.schedule_id = b.schedule_id group by time_slots;
```

Time Slot	Sum(Onboard)
Morning	1497
Evening	1300
Afternoon	789
Night	770

-- 1.5 Demand by specific route and timeslot

To find the utilization of the bus in different slots of the day on a route, first we find the utilization for the bus at different times and routes joining the rides and the bus table and using windows function. The rides table captures the real time data while the schedule table provides the weekdays and the rides information. We can find the data for offboard and onboard for each ride and can then divide by the bus capacity from the bus table which will let us know the current capacity on a specific route. The time slots have been divided into morning, afternoon, evening and night for every three hours in the day for which we can find the busiest slot on a specific route.



```
select time_slots , c.route_short_name , SUM(onboard),
    AVG(onboard) from
(SELECT (CASE
        WHEN HOUR(CAST(b.start_time AS TIME)) >= 6 AND HOUR(CAST(b.start_time AS TIME)) < 11 THEN 'Morning'
        WHEN HOUR(CAST(b.start_time AS TIME)) >= 11 AND HOUR(CAST(b.start_time AS TIME)) < 14 THEN'Afternoon'
        WHEN HOUR(CAST(b.start_time AS TIME)) >= 14 AND HOUR(CAST(b.start_time AS TIME)) < 17 THEN 'Evening'
        WHEN HOUR(CAST(b.start_time AS TIME)) >= 17 AND HOUR(CAST(b.start_time AS TIME)) < 22 THEN 'Night'
        ELSE 'late night'
    END) AS time_slots, b.route_id, onboard
FROM rides a LEFT JOIN schedules b ON a.schedule_id = b.schedule_id ) b
            inner join route c ON b.route_id=c.route_id
GROUP BY time_slots, b.route_id;
```

Time_slot	Route	Sum(onboard)	Avg(onboard)
Night	24	130	3.7143
Evening	24	127	3.6286
Morning	24	123	3.5143
Evening	4B	219	3.1286
Morning	15	95	2.7143
Afternoon	24	281	2.6762
Evening	4A	775	2.4603
Morning	6A	250	2.381
Night	15	83	2.3714
Night	35	73	2.0857
Night	4A	362	2.0686
Morning	4A	894	1.9648
Morning	6B	135	1.9286
Afternoon	6A	66	1.8857
Afternoon	4A	380	1.8095
Afternoon	6B	62	1.7714
Night	6A	122	1.7429
Evening	35	122	1.7429
Evening	28	57	1.6286

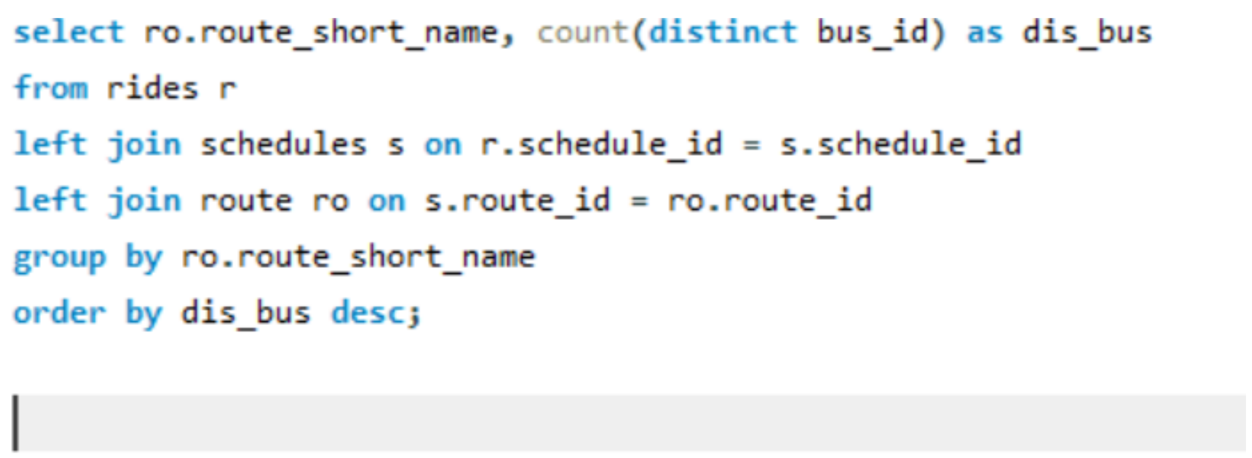
Observation from Demand:

- With total onboard of 2,411 for one week, 4A is the busiest route while 28 is the route least used.
- Tuesday mornings and Monday afternoons are the busiest day of the week while weekends have least number of passengers .
- Tuesday is the busiest day of the week for route 4A.
- Morning and evening are the busiest in a day.
- Route 24 at night has the highest average number of passengers per stops.
- Lowest average for a time slot in a route is in the evening for route 35 and 28.

Second, we find the bus allocation and hence, resource allocation for each of these routes on different days.

-- 2.1 Bus allocation by route

To find the number of buses allocated on different routes, we use the tables of rides, route and schedule to find different information like the route name and the buses for those routes. When we get the real-time data from the rides table, we use routes to map it to the specific routes and the buses operating on those routes. Further, we can find the number of buses operating on a route and then we can find how the buses are allocated and if they are being used efficiently.

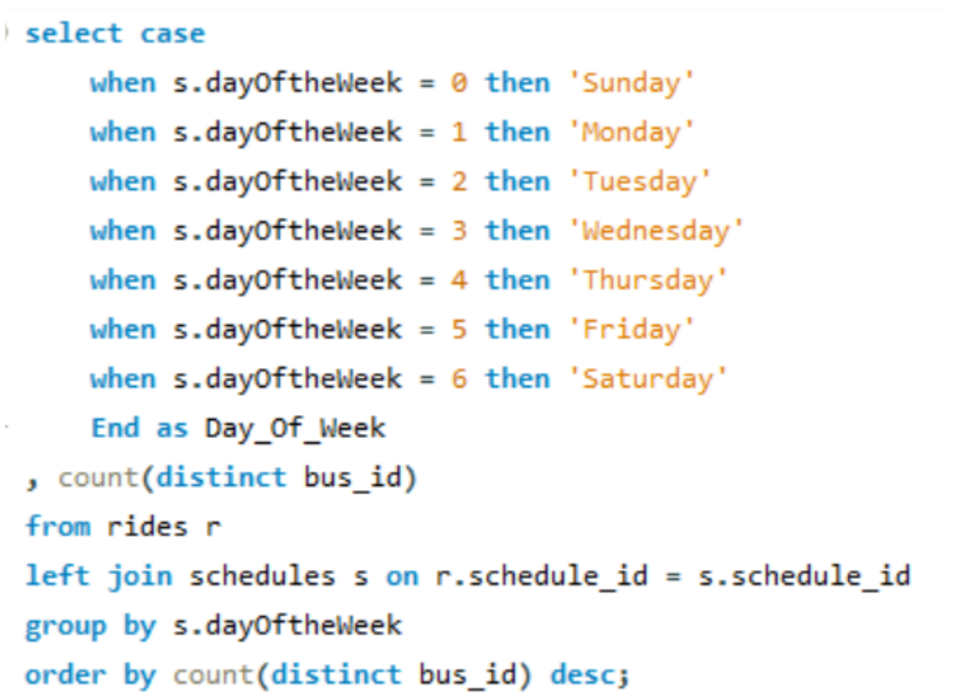


```sql
select ro.route_short_name, count(distinct bus_id) as dis_bus
from rides r
left join schedules s on r.schedule_id = s.schedule_id
left join route ro on s.route_id = ro.route_id
group by ro.route_short_name
order by dis_bus desc;
```

Route	Dis_Bus
4A	3
24	2
4B	2
6B	2
15	1
28	1
35	1
6A	1

-- 2.2 Bus allocation by day of week

We can find the number of buses getting used for each weekday by using both the rides and the schedule table. From rides table, we can find the real time information for each ride daily and then we can use that information to find the buses through the schedule table. We used the schedule table to map the usage of resources for each ride from the rides table in a day.



```
select case
    when s.dayOftheWeek = 0 then 'Sunday'
    when s.dayOftheWeek = 1 then 'Monday'
    when s.dayOftheWeek = 2 then 'Tuesday'
    when s.dayOftheWeek = 3 then 'Wednesday'
    when s.dayOftheWeek = 4 then 'Thursday'
    when s.dayOftheWeek = 5 then 'Friday'
    when s.dayOftheWeek = 6 then 'Saturday'
    End as Day_Of_Week
, count(distinct bus_id)
from rides r
left join schedules s on r.schedule_id = s.schedule_id
group by s.dayOftheWeek
order by count(distinct bus_id) desc;
```

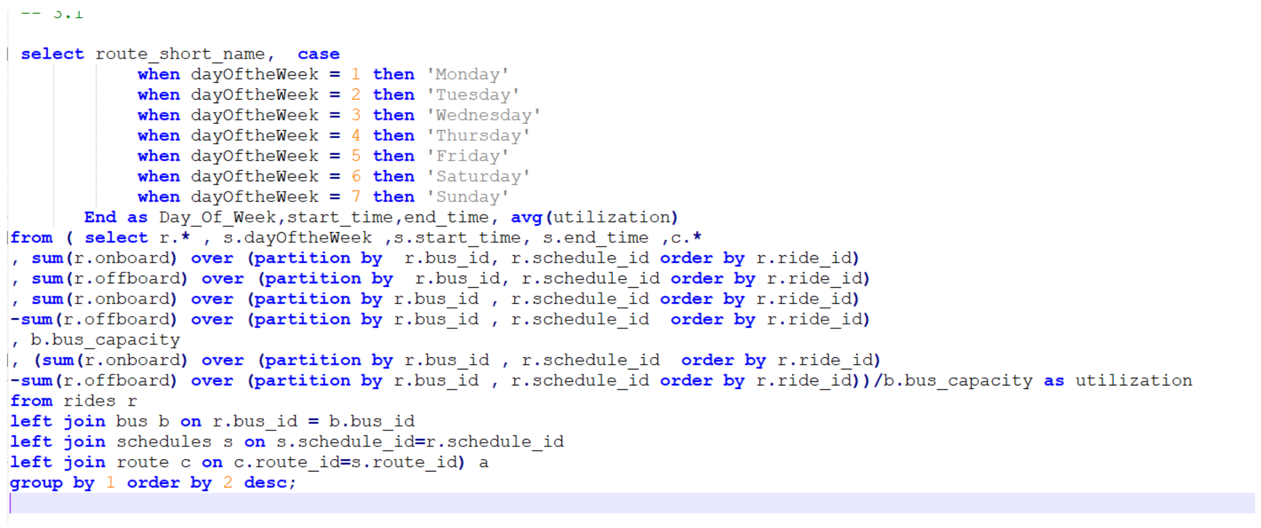
DayOfWeek	Dis_Bus
Wednesday	3
Tuesday	2
Thursday	2
Friday	2
Sunday	2
Monday	1
Saturday	1

Observations from resource allocation:
- 4A route has the highest number of the resource/bus used.
- Most of the buses run on Wednesday .

Finally, we can find the running capacity for the bus with respect to the route and the schedules and can assess the current supply

-- 3.1 Average capacity utilization of a bus by schedule

We can find the current utilization of the capacity of the bus by the schedule timings for each route using this query. For this we use the rides table to capture the real time data for each ride and then use the bus table to find the capacity for each bus for the ride and the schedule of the bus, and the rides for each route. By finding the sum of onboard and offboard and partitioning it by bus and schedule we ensure that the data gets easily partitioned for the different buses and the schedules for each of these buses over a route. We then divide it by the given bus capacity/seat numbers to see the percentage of utilisation that is happening for the bus at a time in a day.

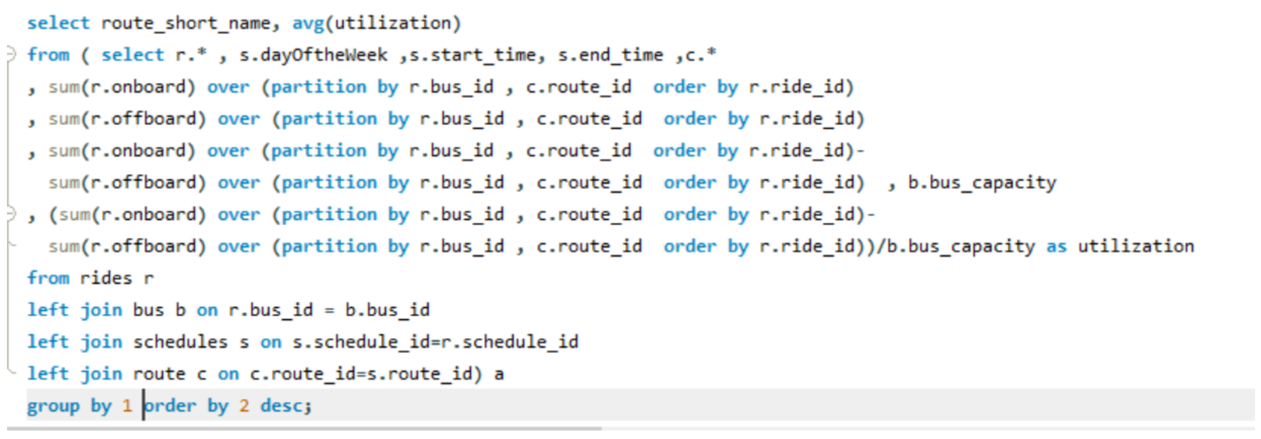

```
-- 3.1

select route_short_name,  case
        when dayOftheWeek = 1 then 'Monday'
        when dayOftheWeek = 2 then 'Tuesday'
        when dayOftheWeek = 3 then 'Wednesday'
        when dayOftheWeek = 4 then 'Thursday'
        when dayOftheWeek = 5 then 'Friday'
        when dayOftheWeek = 6 then 'Saturday'
        when dayOftheWeek = 7 then 'Sunday'
     End as Day_Of_Week,start_time,end_time, avg(utilization)
from ( select r.* , s.dayOftheWeek ,s.start_time, s.end_time ,c.*
, sum(r.onboard) over (partition by  r.bus_id, r.schedule_id order by r.ride_id)
, sum(r.offboard) over (partition by  r.bus_id, r.schedule_id order by r.ride_id)
, sum(r.onboard) over (partition by r.bus_id , r.schedule_id order by r.ride_id)
-sum(r.offboard) over (partition by r.bus_id , r.schedule_id  order by r.ride_id)
, b.bus_capacity
, (sum(r.onboard) over (partition by r.bus_id , r.schedule_id  order by r.ride_id)
-sum(r.offboard) over (partition by r.bus_id , r.schedule_id order by r.ride_id))/b.bus_capacity as utilization
from rides r
left join bus b on r.bus_id = b.bus_id
left join schedules s on s.schedule_id=r.schedule_id
left join route c on c.route_id=s.route_id) a
group by 1 order by 2 desc;
```

Route_short_name	DayOfWeek	StartTime	EndTime	Avg Utilization
4B	Monday	16:00:00	17:00:00	0.47436714
24	Monday	13:00:00	14:00:00	0.40812143
4A	Monday	9:00:00	10:00:00	0.29228035
6B	Thursday	9:00:00	10:00:00	0.2596619
6A	Monday	9:00:00	10:00:00	0.20265619
15	Monday	10:00:00	11:00:00	0.16133714
35	Wednesday	17:00:00	18:00:00	0.1221219
28	Friday	15:00:00	16:00:00	0.12016857

-- 3.2 Average utilization of the bus by route

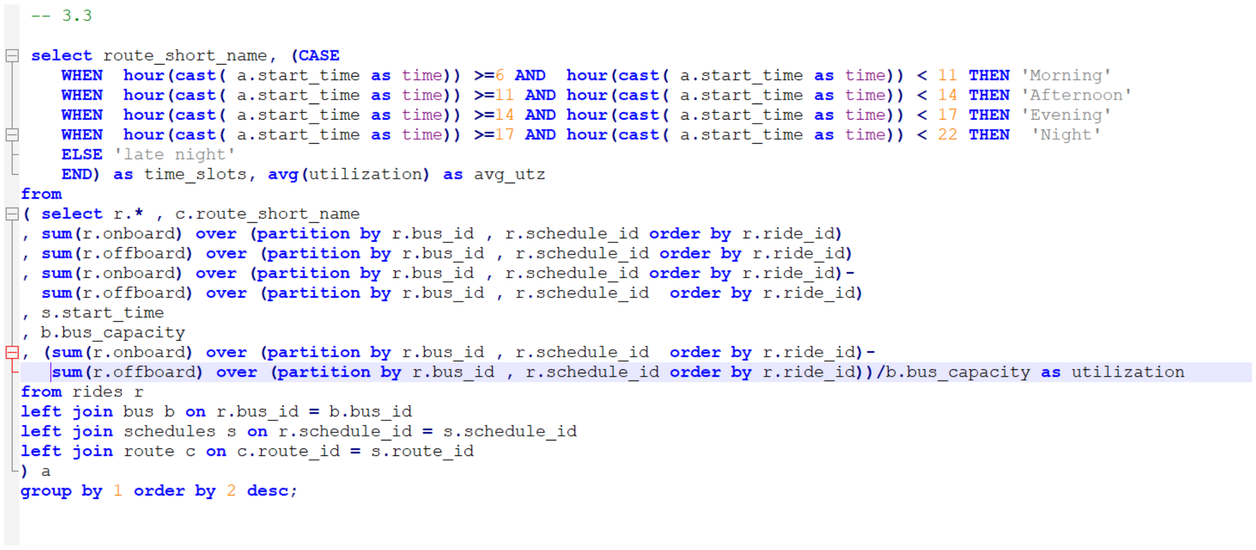
For each route, we can find the capacity utilization using the footfall in each ride through the sum of onboard and sum of offboard and then mapping it to each bus. The bus capacity helps us to determine the exact utilization by route. We can use this information to find the busiest routes. When we join the tables bus, schedule and route we can exactly map buses to their route and their individual schedules. So, the utilization can then get calculated easily for each route and for each bus.


```
select route_short_name, avg(utilization)
from ( select r.* , s.dayOftheWeek ,s.start_time, s.end_time ,c.*
, sum(r.onboard) over (partition by r.bus_id , c.route_id  order by r.ride_id)
, sum(r.offboard) over (partition by r.bus_id , c.route_id  order by r.ride_id)
, sum(r.onboard) over (partition by r.bus_id , c.route_id  order by r.ride_id)-
  sum(r.offboard) over (partition by r.bus_id , c.route_id  order by r.ride_id)  , b.bus_capacity
, (sum(r.onboard) over (partition by r.bus_id , c.route_id  order by r.ride_id)-
  sum(r.offboard) over (partition by r.bus_id , c.route_id  order by r.ride_id))/b.bus_capacity as utilization
from rides r
left join bus b on r.bus_id = b.bus_id
left join schedules s on s.schedule_id=r.schedule_id
left join route c on c.route_id=s.route_id) a
group by 1 order by 2 desc;
```

Route_short_name	Utilization
24	0.59930048
4B	0.47436714
6A	0.47226952
4A	0.44201524
6B	0.41652762
15	0.22016429
35	0.22015905
28	0.12016857

-- 3.3 Capacity utilisation of bus by time slots and route

To find the utilization of a bus by timeslots in a day we define it as the percentage of total number of passengers to the bus capacity within those schedules. We can divide the start and the stop time into four different timeslots by three hours into morning, afternoon, evening and night. With the sum of onboard and offboard passengers that happen over a route we can find the total capacity of a bus for a schedule using the rides data for the different slots. The rides table captures the rides information which can then be used to identify each bus load and when we divide it with the bus capacity from the buses table, we get to find how the passengers can be using it in each time slot of the day which can determine the utilization.



```sql
 -- 3.3

 select route_short_name, (CASE
    WHEN  hour(cast( a.start_time as time)) >=6 AND  hour(cast( a.start_time as time)) < 11 THEN 'Morning'
    WHEN  hour(cast( a.start_time as time)) >=11 AND hour(cast( a.start_time as time)) < 14 THEN 'Afternoon'
    WHEN  hour(cast( a.start_time as time)) >=14 AND hour(cast( a.start_time as time)) < 17 THEN 'Evening'
    WHEN  hour(cast( a.start_time as time)) >=17 AND hour(cast( a.start_time as time)) < 22 THEN  'Night'
    ELSE 'late night'
    END) as time_slots, avg(utilization) as avg_utz
from
( select r.* , c.route_short_name
, sum(r.onboard) over (partition by r.bus_id , r.schedule_id order by r.ride_id)
, sum(r.offboard) over (partition by r.bus_id , r.schedule_id order by r.ride_id)
, sum(r.onboard) over (partition by r.bus_id , r.schedule_id order by r.ride_id)-
  sum(r.offboard) over (partition by r.bus_id , r.schedule_id  order by r.ride_id)
, s.start_time
, b.bus_capacity
, (sum(r.onboard) over (partition by r.bus_id , r.schedule_id  order by r.ride_id)-
   sum(r.offboard) over (partition by r.bus_id , r.schedule_id order by r.ride_id))/b.bus_capacity as utilization
from rides r
left join bus b on r.bus_id = b.bus_id
left join schedules s on r.schedule_id = s.schedule_id
left join route c on c.route_id = s.route_id
) a
group by 1 order by 2 desc;
```

Route_short_name	Time Slot	Utilization
4B	Evening	0.47436714
24	Afternoon	0.40812143
4A	Morning	0.29228035
6B	Morning	0.2596619
6A	Morning	0.20265619
15	Morning	0.16133714

35	**Night**	**0.1221219**
28	**Evening**	**0.12016857**

Observations:

- 24 is the highest utilized route while 28 is the least.
- 4B has the highest utilization in the evening, 24 in the afternoon, 4A in the morning and 35 at the night.
- Highest utilization in an hour for a day of the week in a route if for 4B on Monday between 4:00 PM to 5:00 PM
- Least utilization in an hour for a day of week in a route is for 28 on Friday between 3:00 PM to 4:00 PM

Actionable insights:

This data can help CityBus decide if they need to change the frequency of a route during a particular time of the day. Through the data from the queries, we can make the following recommendations:

- Monday afternoons and Tuesday mornings are the busiest with a footfall of 989 people and 936 respectively. So, we could add more buses in these time slots.
- The bus capacity utilization for Thursdays are very low (25%) and hence few buses can be reallocated to Monday with higher capacity utilization (40%).
- The frequency can be reduced for route-28 as the capacity utilization is very low (12%) while route-4B can be run more times in a day owing to higher capacity utilization (47%)

Financial Impact:

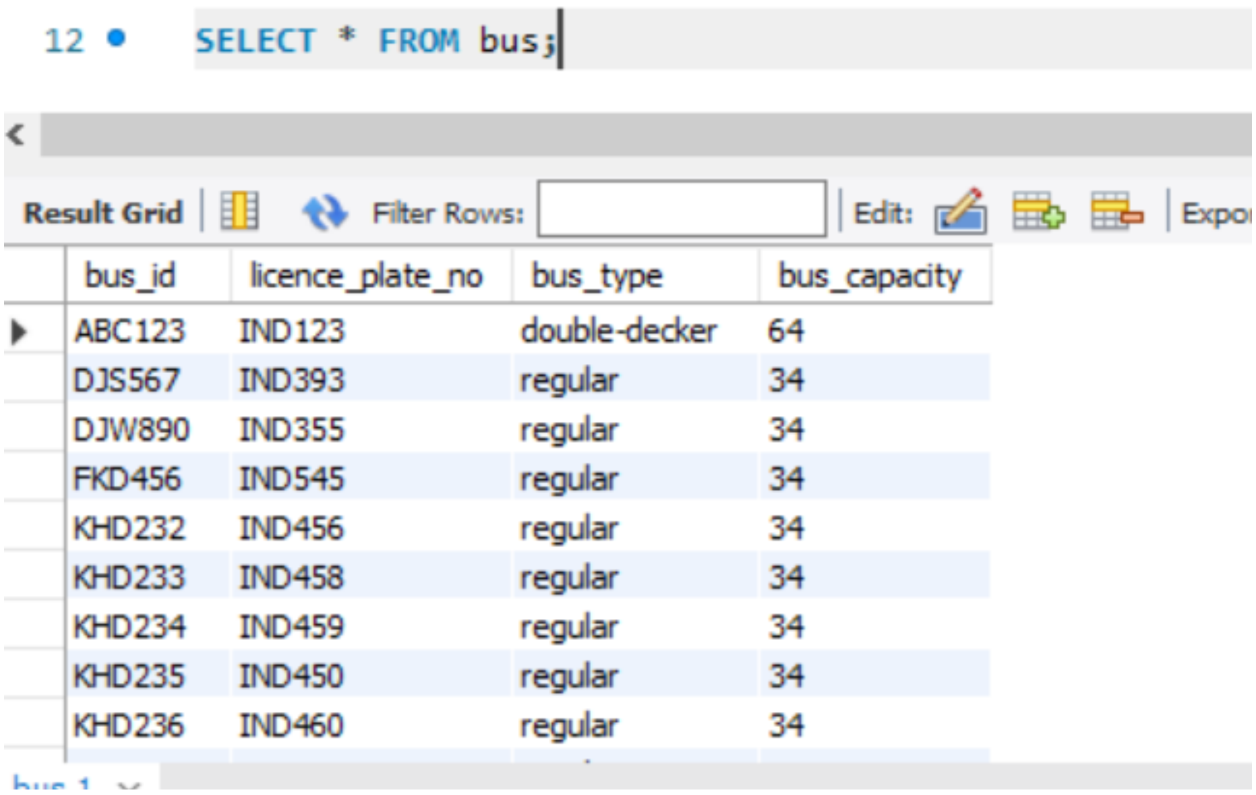
Implementing the solution from the route optimization and implementing the actionable insights, Citybus has the potential to increase the average bus capacity utilization from 30% to 50%.

Using the costs data (Wages for bus drivers, Fuel costs and Bus maintenance costs) taken from Citybus Finance team and estimating the revenues using the utilization data (average price per ride is taken as $1), we could achieve reduction in costs (as percentage of revenues) from 70% to 55%, a reduction of 15%, due to the increased bus capacity utilization.

Appendix:

Snippets from our final tables:

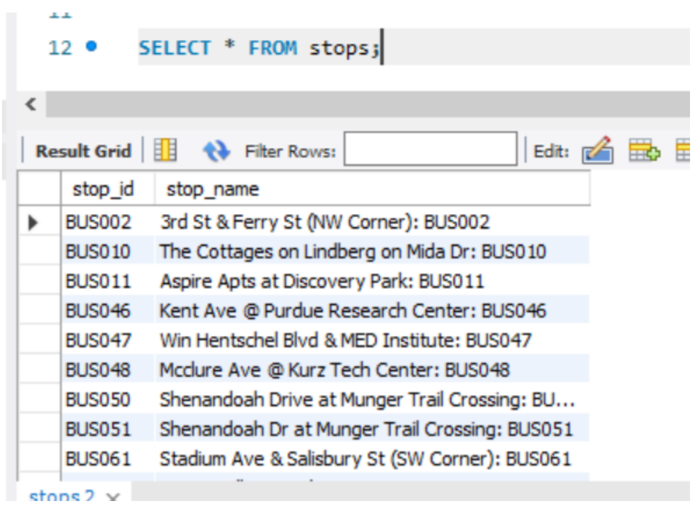
1. Bus:



```
SELECT * FROM bus;
```

bus_id	licence_plate_no	bus_type	bus_capacity
ABC123	IND123	double-decker	64
DJS567	IND393	regular	34
DJW890	IND355	regular	34
FKD456	IND545	regular	34
KHD232	IND456	regular	34
KHD233	IND458	regular	34
KHD234	IND459	regular	34
KHD235	IND450	regular	34
KHD236	IND460	regular	34

2. Stops:



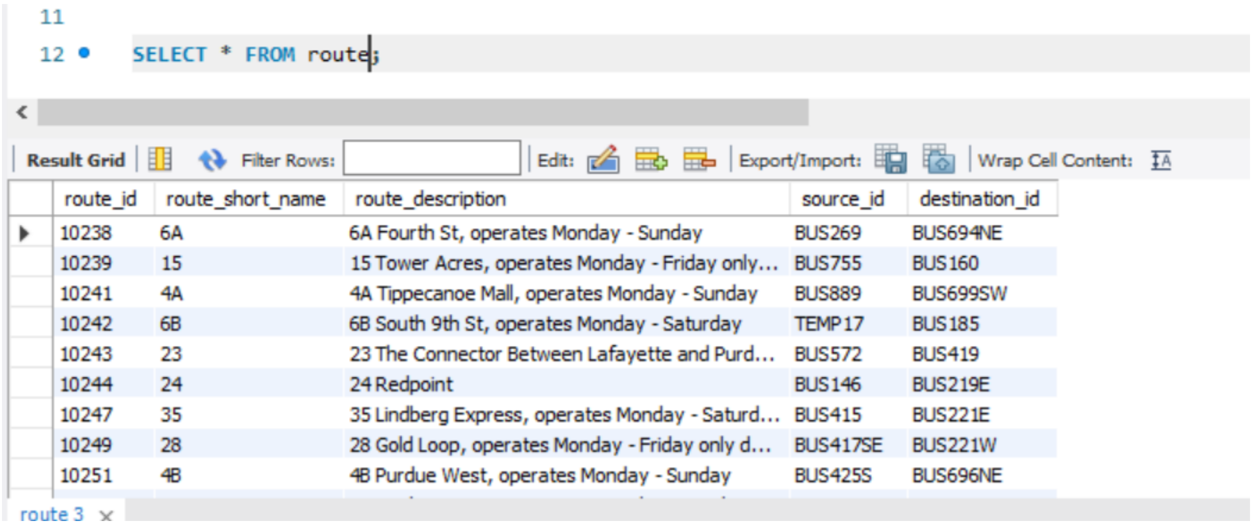
```
SELECT * FROM stops;
```

Result Grid | Filter Rows: | Edit:

	stop_id	stop_name
▶	BUS002	3rd St & Ferry St (NW Corner): BUS002
	BUS010	The Cottages on Lindberg on Mida Dr: BUS010
	BUS011	Aspire Apts at Discovery Park: BUS011
	BUS046	Kent Ave @ Purdue Research Center: BUS046
	BUS047	Win Hentschel Blvd & MED Institute: BUS047
	BUS048	Mcclure Ave @ Kurz Tech Center: BUS048
	BUS050	Shenandoah Drive at Munger Trail Crossing: BU...
	BUS051	Shenandoah Dr at Munger Trail Crossing: BUS051
	BUS061	Stadium Ave & Salisbury St (SW Corner): BUS061

stops 2 ×

3. Route:



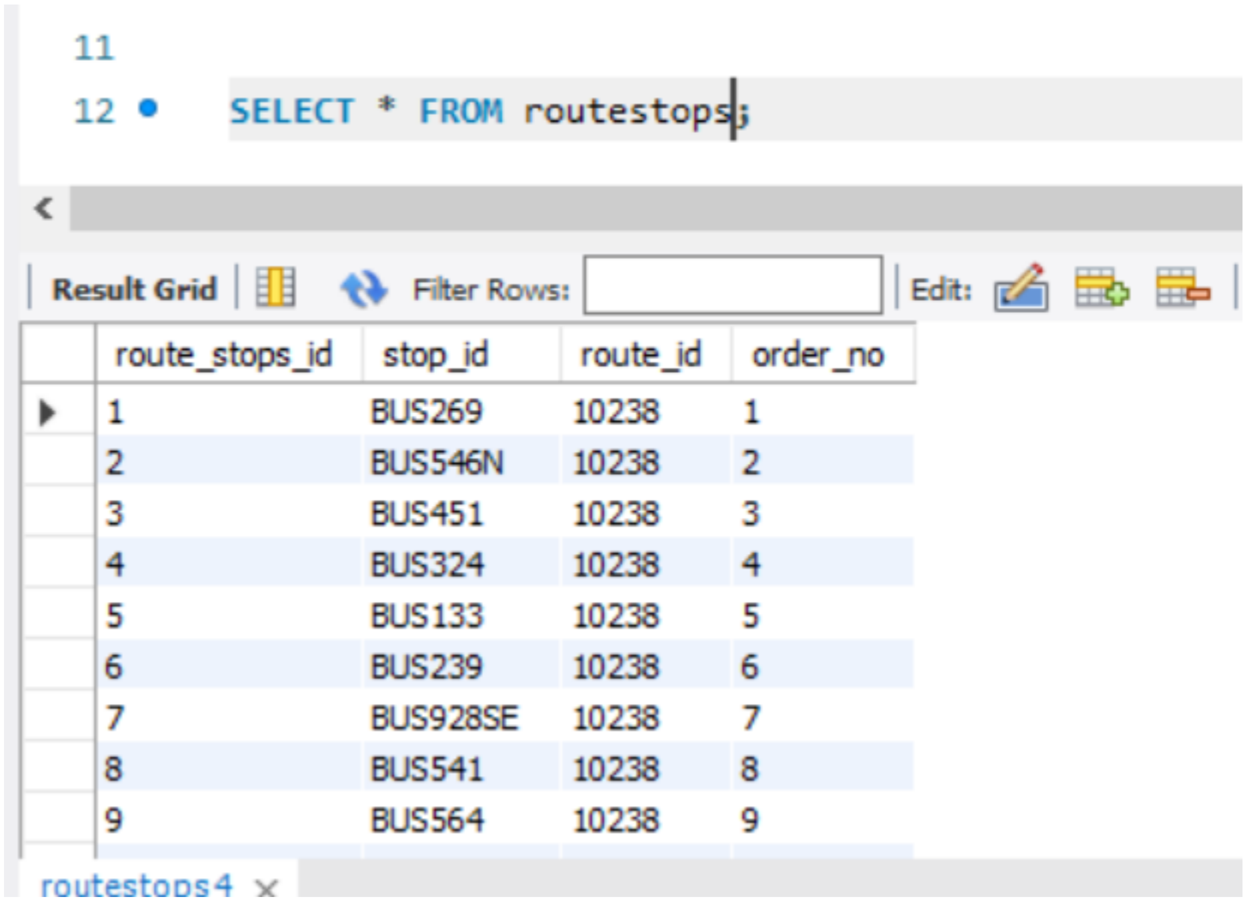
```
SELECT * FROM route;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	route_id	route_short_name	route_description	source_id	destination_id
▶	10238	6A	6A Fourth St, operates Monday - Sunday	BUS269	BUS694NE
	10239	15	15 Tower Acres, operates Monday - Friday only...	BUS755	BUS160
	10241	4A	4A Tippecanoe Mall, operates Monday - Sunday	BUS889	BUS699SW
	10242	6B	6B South 9th St, operates Monday - Saturday	TEMP17	BUS185
	10243	23	23 The Connector Between Lafayette and Purd...	BUS572	BUS419
	10244	24	24 Redpoint	BUS146	BUS219E
	10247	35	35 Lindberg Express, operates Monday - Saturd...	BUS415	BUS221E
	10249	28	28 Gold Loop, operates Monday - Friday only d...	BUS417SE	BUS221W
	10251	4B	4B Purdue West, operates Monday - Sunday	BUS425S	BUS696NE

route 3 ×

4. Routestops:



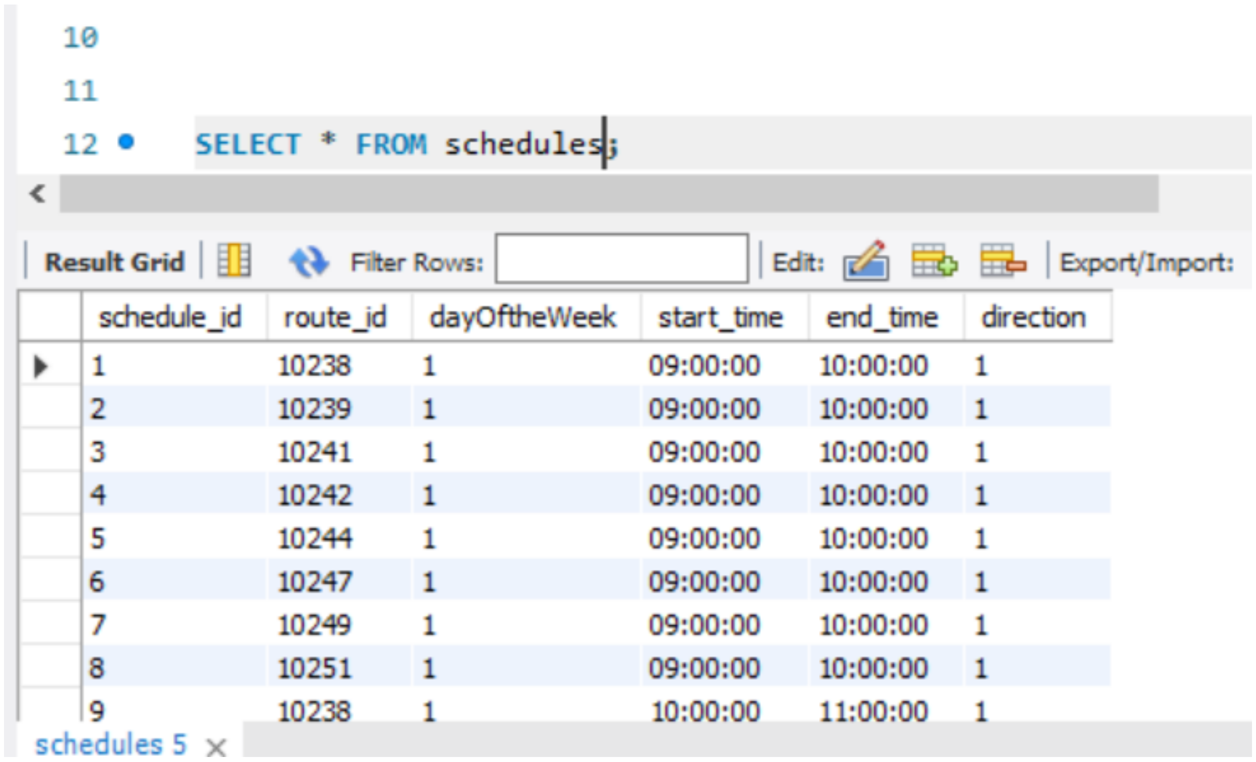
```
SELECT * FROM routestops;
```

Result Grid | Filter Rows: | Edit:

route_stops_id	stop_id	route_id	order_no
1	BUS269	10238	1
2	BUS546N	10238	2
3	BUS451	10238	3
4	BUS324	10238	4
5	BUS133	10238	5
6	BUS239	10238	6
7	BUS928SE	10238	7
8	BUS541	10238	8
9	BUS564	10238	9

routestops4

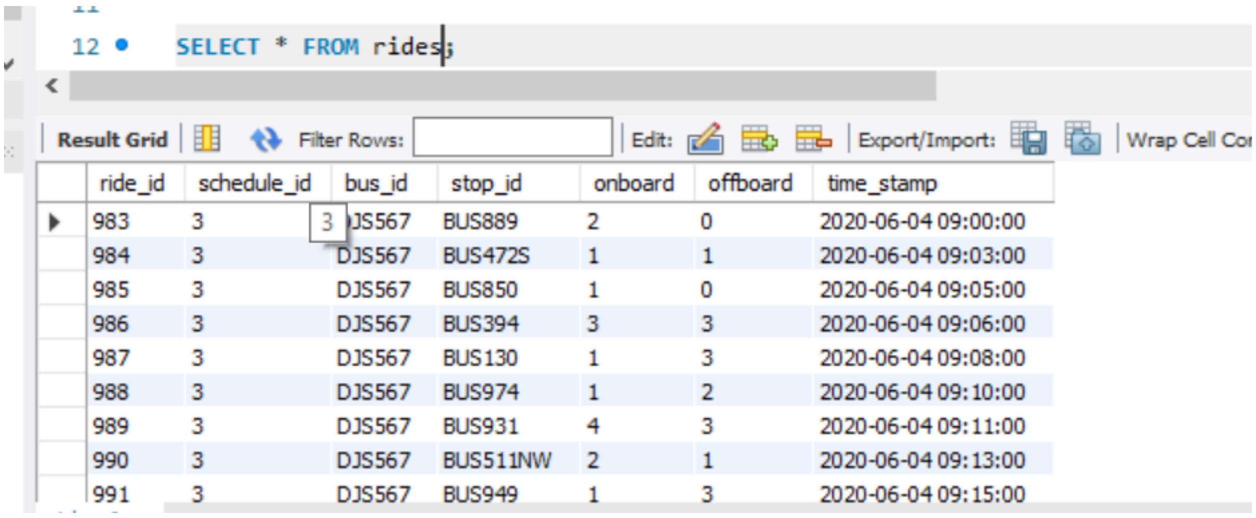
5. Schedules:



```
SELECT * FROM schedules;
```

schedule_id	route_id	dayOftheWeek	start_time	end_time	direction
1	10238	1	09:00:00	10:00:00	1
2	10239	1	09:00:00	10:00:00	1
3	10241	1	09:00:00	10:00:00	1
4	10242	1	09:00:00	10:00:00	1
5	10244	1	09:00:00	10:00:00	1
6	10247	1	09:00:00	10:00:00	1
7	10249	1	09:00:00	10:00:00	1
8	10251	1	09:00:00	10:00:00	1
9	10238	1	10:00:00	11:00:00	1

6. Rides:



```
SELECT * FROM rides;
```

ride_id	schedule_id	bus_id	stop_id	onboard	offboard	time_stamp
983	3	3 JS567	BUS889	2	0	2020-06-04 09:00:00
984	3	DJS567	BUS472S	1	1	2020-06-04 09:03:00
985	3	DJS567	BUS850	1	0	2020-06-04 09:05:00
986	3	DJS567	BUS394	3	3	2020-06-04 09:06:00
987	3	DJS567	BUS130	1	3	2020-06-04 09:08:00
988	3	DJS567	BUS974	1	2	2020-06-04 09:10:00
989	3	DJS567	BUS931	4	3	2020-06-04 09:11:00
990	3	DJS567	BUS511NW	2	1	2020-06-04 09:13:00
991	3	DJS567	BUS949	1	3	2020-06-04 09:15:00