

**Change To 'DATE' type value from 'TEXT' -->>**

**SQL>** ALTER TABLE ola\_data ADD pickup\_dt DATE;

**SQL>** UPDATE ola\_data SET  
pickup\_dt = STR\_TO\_DATE(pickup\_date, '%d-%m-%Y');

## Change String to TIME -->>

**SQL>** UPDATE ola\_data SET pickup\_tm = CAST(pickup\_time AS TIME) ;

The screenshot shows a database management tool interface. On the left, a tree view displays the database structure, including a table named `ola_data` with columns `Booking_id`, `Booking_type`, `Booking_mode`, `Driver_number`, `Service_status`, `Status`, `Fare`, `Distance`, `Confirmed_at`, `pickup_dt`, and `pickup_tm`. The `pickup_tm` column is highlighted with a red box. In the center, the SQL statement `UPDATE ola_data SET pickup_tm = CAST(pickup_time AS TIME) ;` is entered in a text area, also highlighted with a red box. Below the text area, a table shows the results of the update operation. The table has columns `er_number`, `Service_status`, `Status`, `Fare`, `Distance`, `Confirmed_at`, `pickup_dt`, and `pickup_tm`. The data rows show various service statuses and pickup times. On the right, a 'Result Grid' button is visible. At the bottom, an 'Output' pane shows the execution log, including the SQL statement and the number of rows affected.

er_number	Service_status	Status	Fare	Distance	Confirmed_at	pickup_dt	pickup_tm
	done	6	918	61	14-11-2013 09:53	2013-11-15	04:45:00
	cancelled	4	895	60	20-11-2013 22:36	2013-11-21	08:15:00
	done	6	860	57	09-11-2013 12:48	2013-11-09	16:00:00
	done	6	331	22	11-11-2013 07:19	2013-11-11	09:00:00
	done	6	422	20	21-10-2013 20:00	2013-11-01	06:00:00

ola\_data 3    ola\_data 11 x

Output

Action Output

#	Time	Action	Message
✓ 11	11:21:06	SELECT CAST(pickup_time' AS TIME) AS ConvertedTime LIMIT 0, 500	1 row(s) returned
✓ 12	11:21:34	SELECT CAST(pickup_time' AS TIME) from ola_data LIMIT 0, 500	500 row(s) returned
✓ 13	11:24:32	SELECT CAST(pickup_time AS TIME) from ola_data LIMIT 0, 500	500 row(s) returned
✓ 14	11:25:45	UPDATE ola_data SET pickup_tm = CAST(pickup_time AS TIME)	1356 row(s) affected Rows matched: 1356 Changed: 1356 Warnings: 0
✓ 15	11:25:58	select * from ola_data LIMIT 0, 500	500 row(s) returned

## Change STRING to DATETIME -->>

SQL> alter table ola\_data add cnfrmd\_at Datetime ;

SQL> select str\_to\_date(confirmed\_at , '%d-%m-%Y %H:%i') from ola\_data ;

SQL> UPDATE ola\_data

SET cnfrmd\_at = str\_to\_date(confirmed\_at , '%d-%m-%Y %H:%i');

The screenshot displays a database management interface with the following components:

- Schema Tree (Left):** Shows the database structure, including tables like 'ola\_data' and 'ola\_taxi'.
- SQL Editor (Top):** Contains the SQL command: `select STR_TO_DATE(pickup_time, '%H:%i:%S') from ola_data ;`
- Result Grid (Center):** Displays the results of the SQL query. The columns shown are: `er_number`, `cnfrmd_at`, `Service_status`, `Status`, `Fare`, `Distance`, `Confirmed_at`, and `pic`. The data rows show various taxi trips with their respective timestamps and statuses.
- Action Output (Bottom):** Shows the execution log of the SQL commands. The actions performed are:
  - 28 11:53:21 `select str_to_date(confirmed_at , '%d-%m-%Y %H:%i') from ola_data LIMIT 0, 500` (500 row(s) returned)
  - 29 11:55:59 `alter table ola_data add cnfrmd_at Datetime` (0 row(s) affected)
  - 30 11:56:46 `UPDATE ola_data SET cnfrmd_at = str_to_date(confirmed_at , '%d-%m-%Y %H:%i')` (1227 row(s) affected, 1356 rows matched, 1227 changed)

1. Find hour of 'pickup' and 'confirmed\_at' time, and make a column of weekday as "Sun, Mon, etc" next to pickup\_datetime

SQL>

```
select hour(pickup_time),  
pickup_date,  
dayname(pickup_dt) `pickup_day`,  
cnfrmd_at  
from ola_data ;
```

```
7 • select hour(pickup_time),  
8 pickup_date,  
9 dayname(pickup_dt) `pickup_day`,  
10 cnfrmd_at  
11 from ola_data ;
```

Result Grid				
Filter Rows: <input type="text"/>				
Export:				
Wrap Cell Content:				
Fetch rows:				
	hour(pickup_time)	pickup_date	pickup_day	cnfrmd_at
▶	14	14-11-2013	Thursday	2013-11-14 12:51:00
	5	21-11-2013	Thursday	2013-11-20 19:08:00
	19	04-11-2013	Monday	2013-11-04 17:59:00
	18	01-11-2013	Friday	2013-11-01 12:08:00
	11	30-11-2013	Saturday	2013-11-29 11:57:00

Result 3 x

1. Make a table with count of bookings with booking\_type = p2p categorized by booking mode as 'phone', 'online', 'app', etc

```
6 • select count(*) from ola_data where booking_type='p2p'
7   and booking_mode in ('phone','online','app') ;
8
9
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

count(*)
253

SQL> select count(\*) from ola\_data where booking\_type='p2p' and booking\_mode in ('phone','online','app') ;

OR--

```
6 • SELECT Booking_mode, Count(*) as Cnt
7   FROM ola_data
8   WHERE booking_type = "p2p"
9   Group by Booking_mode;
10
11
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Booking_mode	Cnt
phone	212
online	31
app	10

Create columns for pickup and drop ZONES (using Localities data containing Zone IDs against each area) and fill corresponding values against pick-area and drop\_area, using Sheet 'Localities'

```
SQL> SELECT
    od.PickupArea ,
    pickup_zone.zone_id ,
    od.DropArea ,
    drop_zone.zone_id
FROM
    ola_data od
JOIN
    localities pickup_zone ON od.PickupArea = pickup_zone.Area
JOIN
    localities drop_zone ON od.DropArea = drop_zone.Area;
```

OR -->>

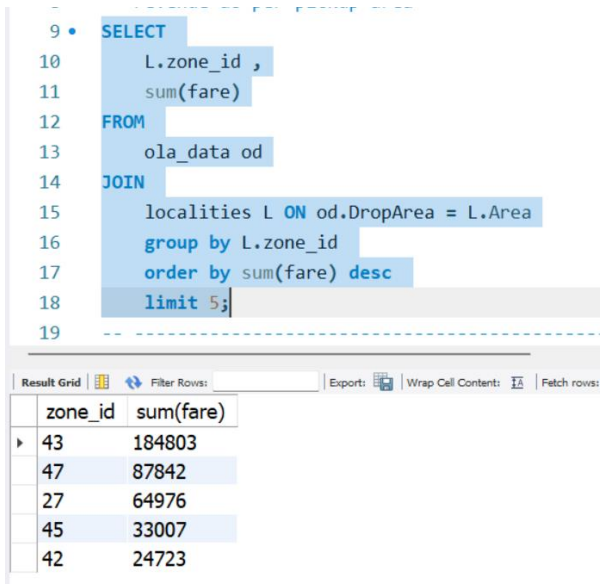
```
SELECT
    od.PickupArea ,
    L1.zone_id ,
    od.DropArea ,
    L2.zone_id
FROM
    ola_data od
JOIN
    localities L1 ON od.PickupArea = L1.Area
JOIN
    localities L2 ON od.DropArea = L2.Area;
```

```
SELECT
    od.PickupArea ,
    pickup_zone.zone_id ,
    od.DropArea ,
    drop_zone.zone_id
FROM
    ola_data od
JOIN
    localities pickup_zone ON od.PickupArea = pickup_zone.Area
JOIN
    localities drop_zone ON od.DropArea = drop_zone.Area;
```

PickupArea	zone_id	DropArea	zone_id
noida sector 122	72	Airport Terminal 1	43
Noida sector 126	72	Airport Terminal 1	43
Noida sector 126	72	Airport Terminal 1	43
crossing republik ghazi...	73	Airport Terminal 1	43
HUDA City Centre, Secto...	60	Airport Terminal 1	43
HUDA City Centre, Secto...	60	Airport Terminal 1	43

-- Top 5 revenue as per pickup area

```
SELECT
    L.zone_id ,
    sum(fare)
FROM
    ola_data od
JOIN
    localities L ON od.DropArea = L.Area
group by L.zone_id
order by sum(fare) desc
limit 5;
```



The screenshot shows a SQL query editor with a query window on the left and a results window on the right. The query window contains the following SQL code:

```
9 • SELECT
10     L.zone_id ,
11     sum(fare)
12 FROM
13     ola_data od
14 JOIN
15     localities L ON od.DropArea = L.Area
16 group by L.zone_id
17 order by sum(fare) desc
18 limit 5;
19
```

The results window displays the output of the query in a table format. The table has two columns: 'zone\_id' and 'sum(fare)'. The results are sorted in descending order of 'sum(fare)'.

zone_id	sum(fare)
43	184803
47	87842
27	64976
45	33007
42	24723

## **TOP 5 Drop Area as per revenue -->>**

```
SQL> SELECT zone_id, Sum(fare) as SumRevenue  
FROM ola_Data as D, Localities as L  
WHERE D.pickuparea = L.Area  
Group By Zone_id  
Order By 2 DESC  
Limit 5;
```

**OR**

## **With Rank -->**

```
with CTE AS (  
    SELECT zone_id, Sum(fare) as SumRevenue  
    FROM ola_Data as D, Localities as L  
    WHERE D.pickuparea = L.Area  
    Group By Zone_id  
    Order By 2 DESC  
    Limit 5)  
select zone_id, SumRevenue ,  
rank() over(order by SumRevenue desc) `rank` from  
CTE ;
```



## Find top 5 drop zones in terms of average revenue

-- Top 5 average revenue as per drop area -->>

```
SELECT zone_id, count(*) `number_of_booking`, round(avg(fare),0) as AvgRevenue
FROM ola_Data as D, Localities as L
WHERE D.pickuparea = L.Area
Group By Zone_id
Order By 2 DESC
Limit 5 ;
```

```
--
22  -- Top 5 average revenue as per drop area -->>
23
24 • SELECT zone_id, count(*) `number_of_booking`, round(avg(fare),0) as AvgRevenue
25 FROM ola_Data as D, Localities as L
26 WHERE D.pickuparea = L.Area
27 Group By Zone_id
28 Order By 2 DESC
29 Limit 5
30
```

Result Grid    Filter Rows: <input type="text"/>   Export:    Wrap Cell Content:    Fetch rows:			
	zone_id	number_of_booking	AvgRevenue
▶	43	267	630
	27	149	593
	47	68	683
	66	63	616
	60	63	636

**Find all unique driver numbers grouped by top 5 pick zones**

```
SQL> SELECT
    pz,
    COUNT(DISTINCT Driver_number) AS unique_driver_count
FROM
    (
        SELECT
            L.Area AS pz,
            COUNT(o.Booking_id) AS total_bookings
        FROM
            ola_data o
        JOIN
            localities L ON o.PickupArea = L.Area
        GROUP BY
            L.Area
        ORDER BY
            total_bookings DESC
        LIMIT 5
    ) top_pickup_zones
JOIN
    ola_data o ON o.PickupArea = top_pickup_zones.pz
GROUP BY
    pz;
```

**Make a hour wise table of bookings for week between Nov01-Nov-07 and highlight the hours with more than average no.of bookings day wise**

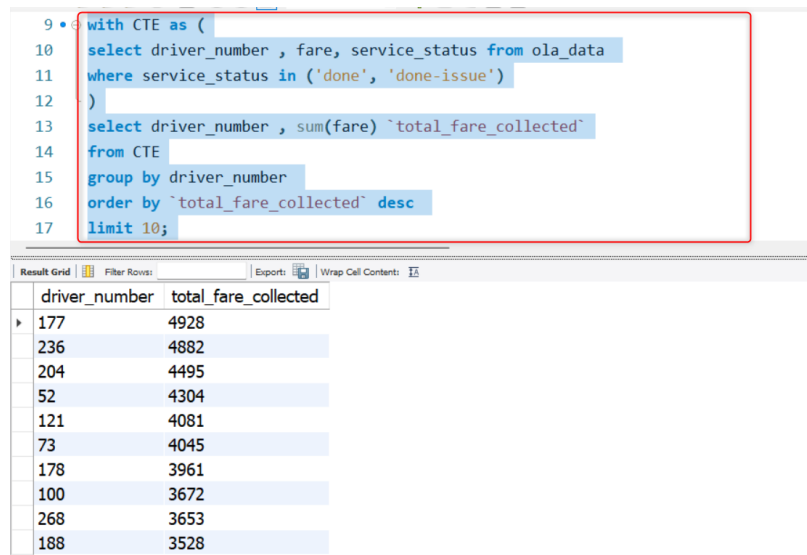
```
SQL>
select hour(pickup_tm) `pickup time`,count(booking_id) `hourly count` from ola_data
where pickup_dt between '2013-11-01' and '2013-11-07'
group by hour(pickup_tm)
order by 1;
```

```
6 • select hour(pickup_tm) `pickup time`,count(booking_id) `hourly count` from ola_data
7 where pickup_dt between '2013-11-01' and '2013-11-07'
8 group by hour(pickup_tm)
9 order by 1;
```

	pickup time	hourly count
▶	0	23
	1	11
	2	11
	3	27
	4	60
	5	70

**Make a list of top 10 driver by driver numbers in terms of fare collected where service\_status is done, done-issue**

```
SQL>
-- find top 10 driver as fare collection
with CTE as (
select driver_number , fare, service_status from ola_data
where service_status in ('done', 'done-issue')
)
select driver_number , sum(fare) `total_fare_collected`
from CTE
group by driver_number
order by `total_fare_collected` desc
limit 10;
```



```
9 • with CTE as (
10   select driver_number , fare, service_status from ola_data
11   where service_status in ('done', 'done-issue')
12 )
13 select driver_number , sum(fare) `total_fare_collected`
14 from CTE
15 group by driver_number
16 order by `total_fare_collected` desc
17 limit 10;
```

driver_number	total_fare_collected
177	4928
236	4882
204	4495
52	4304
121	4081
73	4045
178	3961
100	3672
268	3653
188	3528