SQL 6-Date and Time

Creation of datetime_demo table

```
create table farmers_market.datetime_demo as(
SELECT market_date,
        market_start_time,
        market_end_time,
        CONCAT(market_date, ' ', market_start_time) as market_start_date,
        str_to_date(CONCAT(market_date, ' ', market_start_time), "%Y-%m-%d %h:%i %p") AS
market_start_datetime,
        str_to_date(CONCAT(market_date, ' ', market_end_time), "%Y-%m-%d %h:%i %p") AS
market_end_datetime
          FROM    farmers_market.market_date_info
)
```

Creation of customer_purchases_date table by joining the Customer purchases and market start time, end time from market_date_info

```
create table farmers_market.customer_purchases_date as
(SELECT  c.market_date,
              m.market_start_time,
              m.market_end_time,
              str_to_date(concat(c.market_date, " ", m.market_start_time), '%Y-%m-%d %h:%i %p') as
market_start_datetime,
              str_to_date(concat(c.market_date, " ", m.market_end_time), '%Y-%m-%d %h:%i %p') as
market_end_datetime,
              str_to_date(concat(c.market_date, " ", c.transaction_time), '%Y-%m-%d %H:%i:%s') as
market_date_transaction_time,
              c.transaction_time,
              c.product_id,
              c.vendor_id,
              c.customer_id,
              c.quantity,
              c.cost_to_customer_per_qty
              FROM    farmers_market.customer_purchases c
LEFT JOIN    market_date_info m
              ON c.market_date = m.market_date
)
```

Question: From each market_start_datetime, extract the following:

- day of week,
- month of year,
- year,
- hour and
- minute from the timestamp

```
SELECT
market_start_datetime,
        EXTRACT(YEAR FROM market_start_datetime) AS date_year,
        EXTRACT(DAY FROM market_start_datetime) AS start_day,
        EXTRACT(MONTH FROM market_start_datetime) AS month_of_year,
        EXTRACT(HOUR FROM market_start_datetime) AS hour_of_day,
        EXTRACT(MINUTE FROM market_start_datetime) AS minute_of_time,
        weekday(market_start_date_time) as week_day_no,
        dayname(market_start_date_time) as day_name,
        monthname(market_start_date_time) as month_name
FROM farmers_market.datetime_demo;
```

Extracting the entire date and entire time from the datetime field

```
SELECT market_start_datetime,
     DATE(market_start_datetime) AS mktsrt_date,
     TIME(market_start_datetime) AS mktsrt_time
  FROM farmers_market.datetime_demo
```

Add 30 mins to the start of the market date time

```
SELECT market_start_datetime,
   DATE_ADD(market_start_datetime, INTERVAL 30 MINUTE) AS mktstrt_date_
 plus_30min
 FROM farmers_market.datetime_demo
```

Add/subtract 15 days and 30 days to the start of the market date time using different date function

```
SELECT  market_start_date_time,
        Date_add(market_start_date_time, interval 30 minute) as mkt_date_30_MIN,
        Date_add(market_start_date_time, interval 15 DAY) as mkt_date_15_DAY_add,
        Date_SUB(market_start_date_time, interval 15 DAY) as mkt_date_15_DAY_SUB,
        Date_add(market_start_date_time, interval -30 DAY) as mkt_date_30_day_rev_sub,
        Date_SUB(market_start_date_time, interval -30 DAY) as mkt_date_30_day_rev_add
FROM    farmers_market.datetime_demo
```

Q: Returns the first and last market dates from the **datetime_demo** table, calculates the difference between those two dates

```
 SELECT
     x.first_market,
     x.last_market,
     DATEDIFF(x.last_market, x.first_market) days_first_to_last
  FROM
(
SELECT
        min(market_start_datetime) first_market,
        max(market_start_datetime) last_market
     FROM farmers_market.datetime_demo
)x
```

Calculate the hours and minutes between the market start and end times on each market date.

```
SELECT market_start_datetime,
           market_end_datetime,
         TIMESTAMPDIFF(HOUR, market_start_datetime,
market_end_datetime) AS market_duration_hours,
     TIMESTAMPDIFF(MINUTE, market_start_datetime,
market_end_datetime) AS market_duration_mins
   FROM farmers_market.datetime_demo
```

Question: Let’s say you want to calculate overall sales occurred within the first 30 minutes after the farmer’s market opened

# First Get the row-level transactions within 30 mins from market start time.

```
SELECT  customer_id,
                quantity,
                cost_to_customer_per_qty,
                market_date_transaction_time,
                market_start_datetime,
                date_add(market_start_datetime, interval 30 minute) as mkst_str_30_mins
FROM    farmers_market.customer_purchases_date
where market_date_transaction_time between
                market_start_datetime and date_add(market_start_datetime, interval 30 minute)
order by market_start_datetime
```

# Get the overall sales

```
select sum(quantity * cost_to_customer_per_qty) as overall_30_min_sales
from (
SELECT  customer_id,
                quantity,
                cost_to_customer_per_qty,
                market_date_transaction_time,
                market_start_datetime,
                date_add(market_start_datetime, interval 30 minute) as
mkst_str_30_mins
FROM    farmers_market.customer_purchases_date
where market_date_transaction_time between
                market_start_datetime and date_add(market_start_datetime,
interval 30 minute)
order by market_start_datetime
) x
```

# Get the overall sales: You can do the same in a single query retaining the condition in the where clause.

```
SELECT
sum(quantity * cost_to_customer_per_qty) as overall_30_min_sales
                FROM  farmers_market.customer_purchases_date
where market_date_transaction_time between
                market_start_datetime and
                date_add(market_start_datetime, interval 30 minute)
order by market_start_datetime
```

Question: Let’s say we wanted to get a profile of each farmer’s market customer’s habits over time.

- **Customer’s first purchase**
- **Customer’s last purchase**
- **Total days they made a purchase**
- **how long they are a customer.**
- **Days since their last purchase**

```
SELECT
            customer_id,
            min(market_date) as first_transaction,
            max(market_date) as last_transaction,
            count(*) as total_ transactions,
            count(distinct market_date) as total_market_visits,
            datediff(max(market_date), min(market_date)) as customer_duration,
            datediff(curdate(), max(market_date)) as days_since_lst_purchase

FROM farmers_market.customer_purchases_date
group by customer_id
order by datediff(curdate(), max(market_date))
```

Write a query that gives us the days between each purchase a customer makes

```
SELECT
   customer_id,
   market_date,
   lag(market_date, 1) over (partition by customer_id
order by market_date) as last_purchase,
   datediff(market_date, lag(market_date, 1) over
(partition by customer_id order by market_date)) as
days_from_prev_purchase
FROM farmers_market.customer_purchases_date
```

<u>Let's Extend the query: Find the avg. days it take for the customer between 2 purchases or how long it takes on an avg for a customer to comeback to the market.</u>

```
select customer_id,
         avg(days_from_prev_purchase) from(
SELECT
   customer_id,
   market_date,
   lag(market_date, 1) over (partition by customer_id
order by market_date) as last_purchase,
   datediff(market_date, lag(market_date, 1) over
(partition by customer_id order by market_date)) as
days_from_prev_purchase
FROM farmers_market.customer_purchases_date
) x
group by x.customer_id
```

Question: Assume today’s date is May 31, 2019, and the marketing director of the farmer’s market wants to give infrequent customers(visited only less than 2 days) in the past 30 days an incentive to return to the market in June

```
SELECT
      x.customer_id,
      COUNT(DISTINCT x.market_date) AS total_visits_in_30_days
FROM(
SELECT
      DISTINCT customer_id,
      market_date,
      DATEDIFF('2019-05-31', market_date) as days_before_curr_date
FROM farmers_market.customer_purchases
WHERE DATEDIFF('2019-05-31', market_date) between 0 and 30
)x
GROUP BY x.customer_id
having total_visits_in_30_days <=2
```

Note: Why have we included days between 0 and 30 in the where clause?
The table has data even after May 31st so any days after May 31st will be -1, -2 and so on.
If the condition was only<30 then all those records after May 31st will also be included which is incorrect.