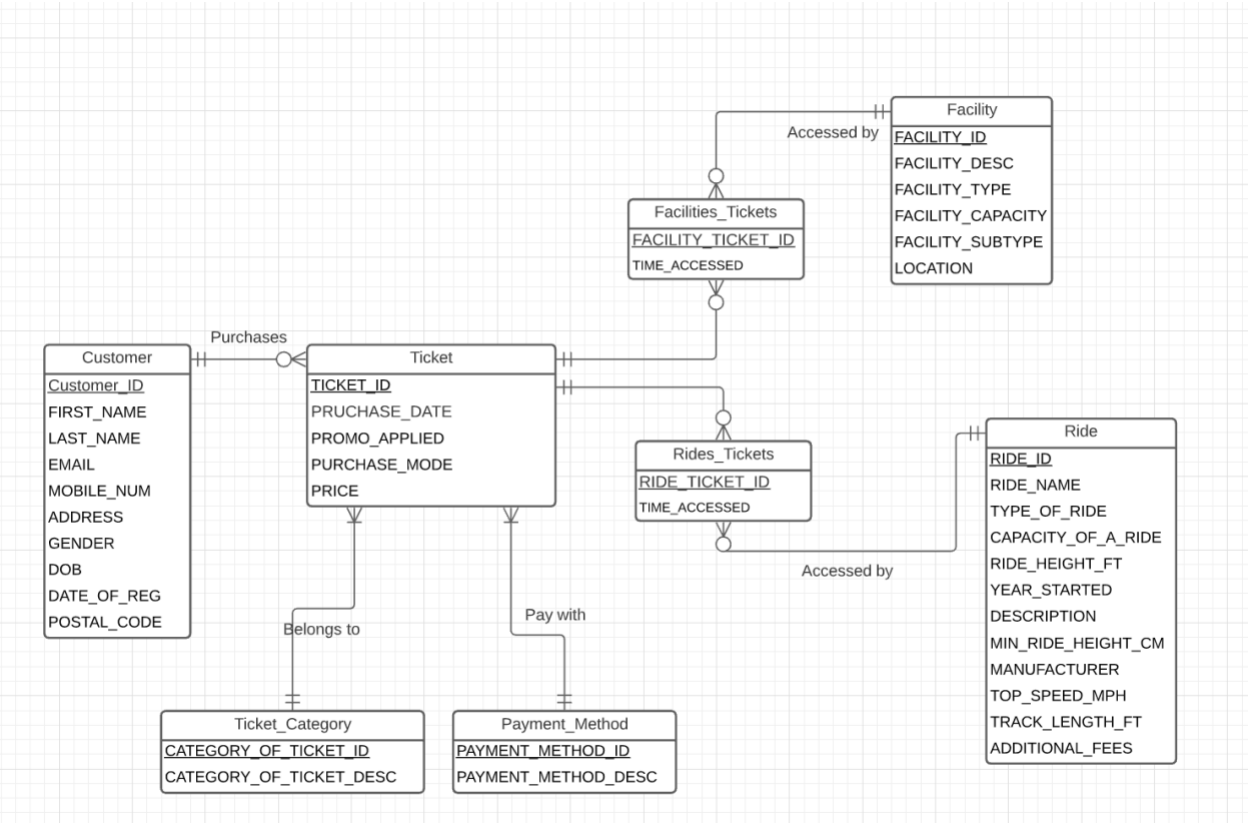
Question 1 to 4 (No external data)

Ziye Zhang 260766101

ERD



Some of Manipulations to populate tables

- Move out Ticket Category and Payment Method
- One customer in Ticket table does not exist in customer table. Corrected it.
- Convert all dates to SQL format, that is yyyy-mm-dd
- Move out TicketID , timestamp from Facility and sorted into distinct ones
- Create associative tables Rides_Tickets and Facilities_Tickets and include timestamp
- Delete an empty column in Ride table, turn brackets of attributes into underscores

Logical Model

Customer (**CUSTOMER_ID**, FIRST_NAME, LAST_NAME, EMAIL, MOBILE_NUM, ADDRESS, GENDER, DOB, DATE_OF_REG, POSTAL_CODE)

PAYMENT_METHOD (**PAYMENT_METHOD_ID**, PAYMENT_METHOD_DESC)

TICKET_CATEGORY (**CATEGORY_OF_TICKET_ID**, CATEGORY_OF_TICKET_DESC)

Ticket (**TICKET_ID**, CUSTOMER_ID, PURCHASE_DATE, PROMO_APPLIED, CATEGORY_OF_TICKET_ID, PAYMENT_METHOD_ID, PURCHASE_MODE, PRICE)

Facility (**FACILITY_ID**, FACILITY_DESC, FACILITY_TYPE, FACILITY_CAPACITY, FACILITY_SUBTYPE, LOCATION)

Facilities_Tickets (**FACILITY_TICKET_ID**, FACILITY_ID, TICKET_ID, TIME_ACCESSED)

Ride (**RIDE_ID**, RIDE_NAME, TYPE_OF_RIDE, CAPACITY_OF_A_RIDE, RIDE_HEIGHT_FT, YEAR_STARTED, DESCRIPTION, MIN_RIDE_HEIGHT_CM, MANUFACTURER, TOP_SPEED_MPH, TRACK_LENGTH_FT, ADDITIONAL_FEES)

Rides_Tickets (**RIDE_TICKET_ID**, RIDE_ID, TICKET_ID, TIME_ACCESSED)

Additional assumptions of business rules.

- A ticket can access many facilities and can also access zero facility.
- A ticket can access many rides and can also access zero ride.
- A Facility or A Ride can have zero or more accesses.
- A ticket must belong to one and only one category.
- A ticket must belong to one and only one payment method.

Queries

Query: 1
Objective:
Get the number of tickets sold and amount sold, and growth rate for each month in order to figure out the trend, which is how sales fluctuate in a longitudinal way.

Assumption:
August 2020 is not considered as it is not a complete month (recorded up to 2020-08-13)

Code:

```
SELECT
   Year, month, Count, Sales, growth_rate
FROM
   (SELECT
      Year,
         month,
         Count,
         Sales,
         IF(@last_entry = 0, 0, ROUND(((Sales - @last_entry) / @last_entry) * 100, 2))
growth_rate,
         @last_entry:=Sales
   FROM
      (SELECT @last_entry:=0) x, (SELECT
      Year, month, Count, Sales
   FROM
      (SELECT
      YEAR(purchase_date) Year,
         MONTH(purchase_date) Month,
         COUNT(ticket_id) Count,
         SUM(Price) AS Sales
   FROM
      Ticket
   WHERE
      purchase_date >= '2019-11-01'
         AND purchase_date <= '2020-07-31'
   GROUP BY YEAR(purchase_date) , MONTH(purchase_date)) temp) y) temp2;
```

Output screenshot:

Year	month	Count	Sales	growth_rate
2019	11	89	28130	0.00
2019	12	123	39850	41.66
2020	1	111	29670	-25.55
2020	2	101	30810	3.84
2020	3	109	31370	1.82
2020	4	113	33690	7.40
2020	5	107	36290	7.72
2020	6	113	36240	-0.14
2020	7	96	32260	-10.98

Query: 2
Objective:
Get the average number of services (rides & facilities) accessed per ticket to determine how many accesses one ticket can make on average.

Assumption:
Services include both rides and facilities.

Code:

```
SELECT
  SUM(num_service) / COUNT(Ticket_ID) AS avg_num_access
FROM
  (SELECT
    Ticket_ID, COUNT(Service_ID) AS num_service
  FROM
    (SELECT
    Ticket_ID, Ride_ID AS Service_ID
  FROM
    Rides_Tickets UNION SELECT
    Ticket_ID, Facility_ID
  FROM
    Facilities_Tickets) service
  GROUP BY Ticket_ID) count_per_ID;
```

Output screenshot:

100% 38:29

Result Grid Filter Rows:

avg_num_access
7.6140

Query: 3
Objective:
Get the average age of customers when they came to La Ronde for the first time to be able to let La Ronde focus on attracting and targeting customers in a specific age group.

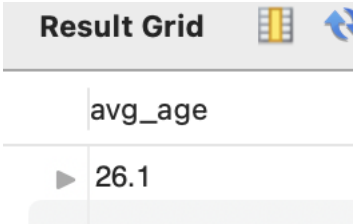
Assumption:
Consider only the customers who have purchased at least a ticket and has at least accessed a ride or facility.

Code:

```
SELECT
  ROUND(AVG(Age), 1) AS avg_age
FROM
  (SELECT
    unique_c.Customer_ID,
      Date_first_entry,
      cc.DOB,
      Date_first_entry - cc.DOB AS Age
  FROM
    (SELECT
    Customer_ID, MIN(Purchase_Date) AS Date_first_entry
  FROM
    (SELECT
    c.Customer_ID, t.Purchase_Date
  FROM
    Customer c
  RIGHT JOIN Ticket t ON c.Customer_ID = t.Customer_ID
  RIGHT JOIN (SELECT
    Ticket_ID, Ride_ID AS Service_ID
  FROM
    Rides_Tickets UNION SELECT
    Ticket_ID, Facility_ID
```

```
  FROM
    Facilities_Tickets) service ON service.Ticket_ID = t.Ticket_ID) ct
  GROUP BY Customer_ID) unique_c
  LEFT JOIN Customer cc ON cc.customer_ID = unique_c.Customer_ID) customer_age;
```

Output screenshot:



avg_age
26.1

Query: 4
Objective:
Find top 5 most loyal customers who spent the most amount, and find how long they have been with La Ronde.

Assumption:
The duration is calculated using the date of first entry when they accessed at least one ride or facility and the last date of the database, which is 2020-08-13.

Code:

```
 SELECT
  customer_loyalty.customer_ID,
  cl.Total_Paid,
  customer_loyalty.Months AS peiod_since_first_visit
FROM
  (SELECT
    Customer_ID,
      MIN(Purchase_Date) AS first_entry,
      STR_TO_DATE('2020-08-13', '%Y-%m-%d') AS Until,
      ROUND(DATEDIFF('2020-08-13', MIN(Purchase_Date)) / 30, 0) Months
  FROM
    (SELECT
    c.Customer_ID, t.Purchase_Date
  FROM
    Customer c
  RIGHT JOIN Ticket t ON c.Customer_ID = t.Customer_ID
  RIGHT JOIN (SELECT
    Ticket_ID, Ride_ID AS Service_ID
  FROM
    Rides_Tickets UNION SELECT
```

```
    Ticket_ID, Facility_ID
  FROM
    Facilities_Tickets) service ON service.Ticket_ID = t.Ticket_ID) ct
  GROUP BY Customer_ID) customer_loyalty
    JOIN
  (SELECT
    customer_ID, SUM(Price) AS Total_paid
  FROM
    Ticket
  GROUP BY Customer_ID) cl ON cl.customer_ID = customer_loyalty.customer_ID
ORDER BY cl.Total_Paid DESC limit 5;
```

Output screenshot:

customer_ID	Total_Paid	peiod_since_first_visit
CD0047	5160	9
CD0124	5030	9
CD0005	4930	9
CD0022	4650	9
CD0103	4540	8

Query: 5
Objective:
Find the top 10 most popular activities (ranked by total number of accesses).

Assumption:
Activities (services) include both rides and facilities.

Code:

```
SELECT
  service.Service_ID, allser.service_name, COUNT(Ticket_ID) as num_accesses
FROM
  (SELECT
    Ticket_ID, Ride_ID AS Service_ID
  FROM
    Rides_Tickets UNION SELECT
    Ticket_ID, Facility_ID
  FROM
    Facilities_Tickets) service
    LEFT JOIN
  (SELECT
    Facility_ID AS service_ID, Facility_desc AS service_name
  FROM
```

```
    Facility UNION SELECT
    Ride_ID, Ride_Name
  FROM
    Ride) allser ON allser.service_ID = service.service_ID
GROUP BY Service_ID , service_Name order by num_accesses desc limit 10;
```

Output screenshot:

Service_ID	service_name	num_accesses
FAC110	Fines Poutines Express	113
R005	Boomerang	110
FAC137	Popcorn & Cie	108
FAC139	Popcorn & Cie	106
R010	Dragon	106
R016	Gravitor	105
R033	Splash	105
FAC128	Au Comptoir Frais	103
R037	Tour de Ville	103
R008	Condor	102

Query: 6
Objective:
Determine if applying additional fees to some rides decreases the popularity (number of accesses) of them.

Code:

```
SELECT
  'With Addtional Fee' AS Category,
  COUNT(Ride_ID) / COUNT(DISTINCT (Ride_ID)) AS avg_visits_per_ride
FROM
  (SELECT
    r.Ride_ID, r.Ride_Name
  FROM
    ride r
  RIGHT JOIN Rides_Tickets rt ON r.Ride_ID = rt.Ride_ID
  WHERE
    Additional_fees = 'Y') with_fees
UNION SELECT
  'Without Addtional Fee',
  COUNT(Ride_ID) / COUNT(DISTINCT (Ride_ID))
FROM
  (SELECT
```

```
    r.Ride_ID, r.Ride_Name
  FROM
    ride r
  RIGHT JOIN Rides_Tickets rt ON r.Ride_ID = rt.Ride_ID
  WHERE
    Additional_fees = 'N') without_fees;
```

Output screenshot:

Category	avg_visits_per_ride
With Addtional Fee	90.0000
Without Addtional Fee	95.4750

Query: 7
Objective:
See if people are excited about promotions and as a result access more rides and facilities.

Code:

```
SELECT
  Promo_Applied,
  COUNT(Ticket_ID) / COUNT(DISTINCT (Ticket_ID)) AS average_access_per_ticket
FROM
  (SELECT
    t.Ticket_ID, t.Promo_Applied
  FROM
    Ticket t
  LEFT JOIN (SELECT
    Ticket_ID, Ride_ID AS Service_ID
  FROM
    Rides_Tickets UNION SELECT
    Ticket_ID, Facility_ID
  FROM
    Facilities_Tickets) service ON t.Ticket_ID = service.Ticket_ID) accesses
GROUP BY Promo_Applied;
```

Output screenshot:

Promo_Applied	average_access_per_ticket
0	7.6323
1	7.5960

Query: 8
Objective:
Compare the favorite service(s) (ride or facility) for customers holding different categories of tickets.

Assumptions:
Include activities that have the same number.

Code:

```
SELECT
   a.Categ_Ticket, a.service_ID, a.Count
FROM
   (SELECT
      category.Category_of_Ticket_Desc AS Categ_Ticket,
         category.Service_ID,
         COUNT(category.Service_ID) AS count
   FROM
      (SELECT
      t.Ticket_ID,
         t.Category_of_Ticket_ID,
         tc.Category_of_Ticket_Desc,
         service.Service_ID
   FROM
      Ticket t
   LEFT JOIN (SELECT
      Ticket_ID, Ride_ID AS Service_ID
   FROM
      Rides_Tickets UNION SELECT
      Ticket_ID, Facility_ID
   FROM
      Facilities_Tickets) service ON t.Ticket_ID = service.Ticket_ID
   LEFT JOIN Ticket_Category tc ON t.Category_of_Ticket_ID = tc.Category_of_Ticket_ID)
category
   GROUP BY category.Category_of_Ticket_Desc , category.Service_ID) a
      INNER JOIN
   (SELECT
      Categ_Ticket, MAX(count) AS maxCount
   FROM
      (SELECT
      category.Category_of_Ticket_Desc AS Categ_Ticket,
         category.Service_ID,
         COUNT(category.Service_ID) AS count
   FROM
      (SELECT
```

```
    t.Ticket_ID,
      t.Category_of_Ticket_ID,
      tc.Category_of_Ticket_Desc,
      service.Service_ID
  FROM
    Ticket t
  LEFT JOIN (SELECT
    Ticket_ID, Ride_ID AS Service_ID
  FROM
    Rides_Tickets UNION SELECT
    Ticket_ID, Facility_ID
  FROM
    Facilities_Tickets) service ON t.Ticket_ID = service.Ticket_ID
  LEFT JOIN Ticket_Category tc ON t.Category_of_Ticket_ID = tc.Category_of_Ticket_ID)
category
  GROUP BY category.Category_of_Ticket_Desc , category.Service_ID) ab
  GROUP BY Categ_Ticket) b ON a.Categ_Ticket = b.Categ_Ticket
    AND a.Count = b.maxCount;
```

Output screenshot:

Categ_Ticket	service_ID	Count
Annual Pass	R005	43
Annual Pass	R007	43
Daily Pass	R037	48
Parking Ticket	FAC103	42

Query: 9
Objective:
See during which period of a day people more frequently access rides and facilities.

Assumption:
There are three time periods: morning 8am-12pm, afternoon 12pm-4pm, night 4pm-8am

Anomalies:
There are timestamps that are happening midnight which do not make sense.

Code:

```
SELECT
  Period, COUNT(Ticket_ID) as Accesses
FROM
  (SELECT
```

```
    Ticket_ID,
      Service_ID,
      CASE -- allocate timestamp to three periods in a day (i.e., morning, afternoon, and
night)
        WHEN Time BETWEEN '08:00:00' AND '12:00:00' THEN 'Morning'
        WHEN Time BETWEEN '12:00:01' AND '16:00:00' THEN 'Afternoon'
        ELSE 'Night'
      END AS Period
  FROM
    (SELECT
    Ticket_ID, Ride_ID AS Service_ID, TIME(Time_Accessed) Time
  FROM
    Rides_Tickets UNION SELECT
    Ticket_ID, Facility_ID, Time_Accessed TIMEONLY
  FROM
    Facilities_Tickets) service) period
GROUP BY Period ORDER BY Accesses DESC;
```

Output screenshot:

Period	Accesses
Night	6650
Afternoon	676
Morning	671

Query: 10
Objective:
Compare school stuff and students with the other kinds of clients to see if they have different favorited rides and facilities.

Assumption:
School stuff and students are the customers whose email ends with 'edu'.

Code:

```
select cc.Customer_type,cc.Service_ID,service.Service_Name from (
(select  'School_customer' as Customer_type,Facility_ID as Service_ID, count(Facility_ID) as
Freq from (
select ct.Customer_ID, ct.Ticket_ID, ft.Facility_ID from(
select c.Customer_ID, t.Ticket_ID from
(select Customer_ID from Customer where Email like '%edu') c left join Ticket t on
t.customer_ID=c.customer_ID) ct
```

```
left join Facilities_Tickets ft on ft.Ticket_ID=ct.Ticket_ID) cft group by Facility_ID order by Freq
desc limit 1 )
Union
(select  'School_customer',Ride_ID,count(Ride_ID) as Freq from (
select ct.Customer_ID, ct.Ticket_ID, rt.Ride_ID from(
select c.Customer_ID, t.Ticket_ID from
(select Customer_ID from Customer where Email like '%edu') c left join Ticket t on
t.customer_ID=c.customer_ID) ct
left join Rides_Tickets rt on rt.Ticket_ID=ct.Ticket_ID) cft group by Ride_ID order by Freq desc
limit 1) Union
(select  'Non_School_customer',Facility_ID, count(Facility_ID) as Freq from (
select ct.Customer_ID, ct.Ticket_ID, ft.Facility_ID from(
select c.Customer_ID, t.Ticket_ID from
(select Customer_ID from Customer where Email not like '%edu') c left join Ticket t on
t.customer_ID=c.customer_ID) ct
left join Facilities_Tickets ft on ft.Ticket_ID=ct.Ticket_ID) cft group by Facility_ID order by Freq
desc limit 1 ) Union
(select  'Non_School_customer',Ride_ID,count(Ride_ID) as Freq from (
select ct.Customer_ID, ct.Ticket_ID, rt.Ride_ID from(
select c.Customer_ID, t.Ticket_ID from
(select Customer_ID from Customer where Email not like '%edu') c left join Ticket t on
t.customer_ID=c.customer_ID) ct
left join Rides_Tickets rt on rt.Ticket_ID=ct.Ticket_ID) cft group by Ride_ID order by Freq desc
limit 1)) cc join -- join to get service name
(SELECT Facility_ID as Service_ID, Facility_Desc AS Service_Name FROM Facility UNION
SELECT Ride_ID,Ride_Name FROM Ride) service on service.Service_ID = cc.service_ID;
```

Output screenshot:

Customer_type	Service_ID	Service_Name
School_customer	FAC110	Fines Poutines Express
School_customer	R033	Splash
Non_School_customer	FAC139	Popcorn & Cie
Non_School_customer	R005	Boomerang