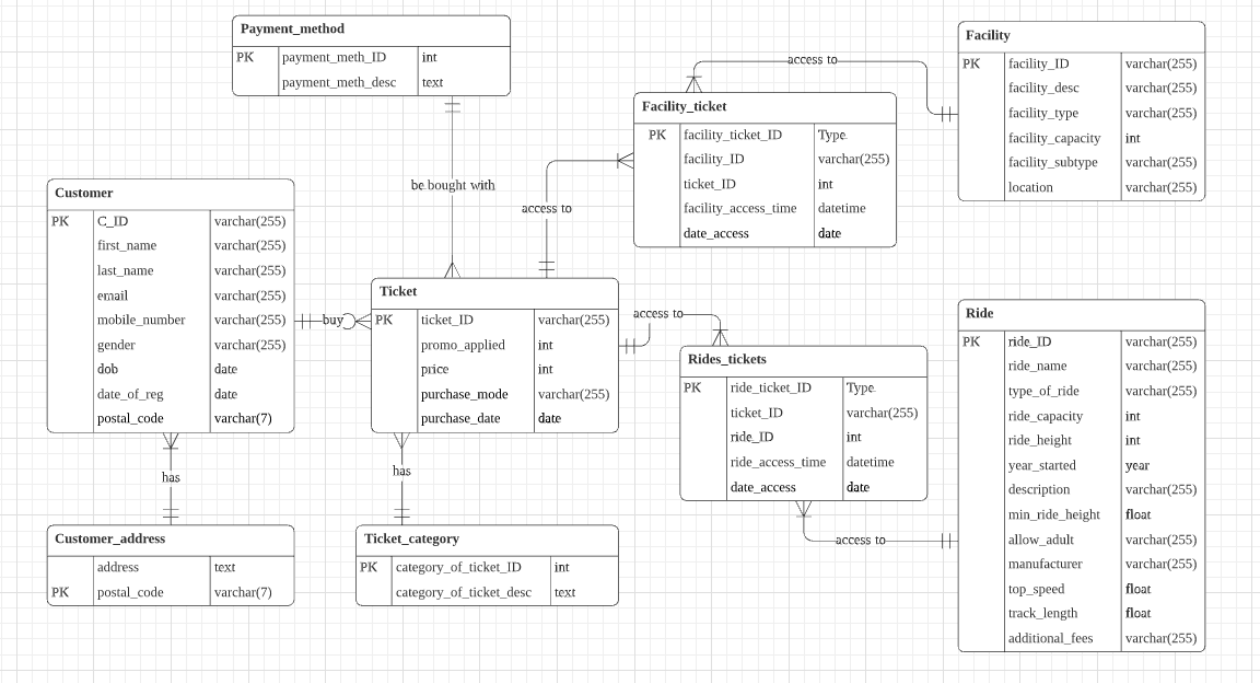
INSY 661

Individual Project – Report 1: La Ronde Data

1. ERD



2. Logical Model

Customer (C_ID, first_name, last_name, email, mobile_number, gender, date_of_reg, dob, postal_code)

Ticket (ticket_ID, promo_applied, price, C_ID, category_of_ticket_ID, payment_meth_ID)

Facility (facility_ID, facility_desc, facility_type, facility_capacity, facility_subtype, location)

Ride (ride_ID, ride_name, type_of_ride, ride_capacity, ride_height, year_started, description, min_height, allow_adult, manufactuerer, top_speed, track_length, additional_fees, allow_adult)

Ticket_category (category_of_tieckt_ID, category_of_ticket_desc)

Payment_method (payment_meth_ID, payment_meth_desc)

Customer_address (postal_code, address)

Ticket_to_ride (ride_ticket_ID, ticket_ID, ride_ID, ride_access_time)

Facility_tickets (facility_ticket_ID, ticket_ID, facility_ID, facility_access_time)

Query 1_____

Objective: compare the total amount of money that customers spent on buying tickets with different payment methods in year 2020. This is to reveal which payment method was most preferred by customers.

Code:

```
SELECT pm.payment_method_desc AS payment_methods, SUM(t.price) AS
total_money_spent

FROM ticket AS t

INNER JOIN payment_method AS pm ON pm.payment_method_ID = t.payment_method_ID

WHERE YEAR(t.purchase_date) = 2020

GROUP BY t.payment_method_ID

ORDER BY SUM(t.price) DESC;
```

Output:

payment_methods	total_money_spent
Cash	62870
Online Payment	62460
Credit Card	59890
Debit Card	56110

Query 2_____

Objective: find the ticket category that generated the most revenue in year 2020 (exclude the revenues generated by tickets that were applied with promo code)

Code:

```
SELECT tc.category_of_ticket_desc AS ticket_type, SUM(t.price) AS revenue

FROM ticket AS t

INNER JOIN ticket_category AS tc ON t.category_of_ticket_ID = tc.category_of_ticket_ID

WHERE YEAR(t.purchase_date) = 2019

AND promo_applied = 1

GROUP BY tc.category_of_ticket_ID

ORDER BY SUM(t.price) DESC;
```

Output:

ticket_type	revenue
Annual pass	28000
Daily Pass	3600
Parking ticket	740

Query 3 _____

Objective: compare the number of tickets of different ticket categories that were purchased via different purchase modes. This query helps La Ronde to know the preference about purchase mode of customers that bought different types of tickets.

Code:

```
SELECT ap.p_mode AS purchase_mode, ap.annual_passes AS annual_pass, pt.parking_tickets
AS parking_ticket, dp.daily_passes AS daily_pass

FROM (

        SELECT t.purchase_mode AS p_mode, COUNT(t.ticket_id) AS annual_passes

        FROM Ticket AS t

        INNER JOIN ticket_category AS tc

        ON tc.category_of_ticket_ID = t.category_of_ticket_ID

        WHERE t.category_of_ticket_ID = 1

        GROUP BY t.purchase_mode) AS ap

INNER JOIN (

    SELECT t2.purchase_mode AS p_mode, COUNT(t2.ticket_id) AS parking_tickets

        FROM Ticket AS t2

        INNER JOIN ticket_category AS tc2

        ON tc2.category_of_ticket_ID = t2.category_of_ticket_ID

        WHERE t2.category_of_ticket_ID = 2

        GROUP BY t2.purchase_mode) AS pt ON pt.p_mode = ap.p_mode

INNER JOIN (

        SELECT t3.purchase_mode AS p_mode, COUNT(t3.ticket_id) AS daily_passes

        FROM Ticket AS t3

        INNER JOIN ticket_category AS tc3

        ON tc3.category_of_ticket_ID = t3.category_of_ticket_ID

        WHERE t3.category_of_ticket_ID = 3

        GROUP BY t3.purchase_mode) AS dp ON dp.p_mode = ap.p_mode;
```

Output:

purchase_mode	annual_pass	parking_ticket	daily_pass
Online	169	135	176
Offline	174	178	168

Query 4_____

Objective: to calculate the proportion of customers of each age intervals. This query allows us to know which age group the main source of customers belong to and adjust La Ronde's entertainment strategy accordingly.

Assumption: I categorized customers into different age intervals to enable queries that compare popularity of facilities and rides among customers of different age intervals.

Kids: 0~15 years old;

Young: 16~35 years old;

Middle: 36~50 years old;

Elder: 51 years old and up.

I added a categorical attribute called "age_invertal" to the Customer table with the following codes:

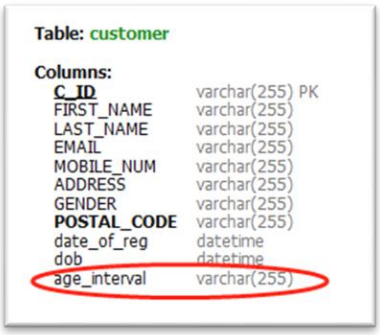
Code:

```
ALTER TABLE Customer
ADD COLUMN age_interval varchar(255);
SET SQL_SAFE_UPDATES = 0;

UPDATE Customer
SET age_interval = 'kid' WHERE YEAR(dob)>=2006;

UPDATE Customer
SET age_interval = 'young' WHERE YEAR(dob)<2006 AND YEAR(dob)>=1986;

UPDATE Customer
SET age_interval = 'middle' WHERE YEAR(dob)<1986 AND YEAR(dob)>=1971;

UPDATE Customer
SET age_interval = 'elder' WHERE YEAR(dob)<1971;

SELECT kid.count/main.total AS kid_proportion,
    young.count/main.total AS young_proportion,
    middle.count/main.total AS middle_proportion,
    elder.count/main.total AS elder_proportion
FROM (
    SELECT COUNT(ticket_ID) AS total
```

```
        FROM Ticket) AS main
    INNER JOIN (
        SELECT COUNT(c.C_ID) AS count
        FROM Ticket AS t
        INNER JOIN Customer AS c ON c.C_ID = t.customer_ID
        WHERE c.age_interval = "kid") AS kid
    INNER JOIN (
        SELECT COUNT(c.C_ID) AS count
        FROM Ticket AS t
        INNER JOIN Customer AS c ON c.C_ID = t.customer_ID
        WHERE c.age_interval = "young") AS young
    INNER JOIN (
        SELECT COUNT(c.C_ID) AS count
        FROM Ticket AS t
        INNER JOIN Customer AS c ON c.C_ID = t.customer_ID
        WHERE c.age_interval = "middle") AS middle
    INNER JOIN (
        SELECT COUNT(c.C_ID) AS count
        FROM Ticket AS t
        INNER JOIN Customer AS c ON c.C_ID = t.customer_ID
        WHERE c.age_interval = "elder") AS elder;
```

Output:

kid_proportion	young_proportion	middle_proportion	elder_proportion
0.2520	0.3860	0.2490	0.1100



Query 5

Objective: to know if young and kid customers from different age groups had similar preferences of rides, i.e. to show the overlaps on young customers' and kids' lists of top 10 preferred rides.

Assumption: one ride_ticket_ID means one visit of a ticket to a ride (customer scans ticket in front of the ride). The popularity of that ride the query uses to rank rides is calculated based on instances of ride_ticket_ID.

Code:

```
SELECT DISTINCT young.names_r AS Same_preference, young.id AS Same_preference_id,
young.young_times AS times_played_by_young, kid.kids_times AS times_played_by_kids
FROM (
    SELECT r.ride_name AS names_r, rt.ride_id AS id, COUNT(rt.ride_id) AS young_times
    FROM Ticket AS t
    INNER JOIN Customer AS c ON t.customer_ID = c.C_ID
    INNER JOIN Rides_tickets AS rt ON rt.ticket_id = t.ticket_id
  INNER JOIN Ride AS r ON r.ride_id = rt.ride_id
    WHERE c.age_interval = 'young'
    GROUP BY rt.ride_id
    ORDER BY COUNT(rt.ride_id) DESC
    LIMIT 10) AS young
INNER JOIN(
    SELECT rt.ride_id AS id, COUNT(rt.ride_id) AS kids_times
    FROM Ticket AS t
    INNER JOIN Customer AS c ON t.customer_ID = c.C_ID
    INNER JOIN Rides_tickets AS rt ON rt.ticket_id = t.ticket_id
    WHERE c.age_interval = 'kid'
    GROUP BY rt.ride_id
    ORDER BY COUNT(rt.ride_id) DESC
    LIMIT 10) AS kid ON kid.id = young.id
ORDER BY young.young_times DESC;
```

Output:

Same_preference	Same_preference_id	times_played_by_young	times_played_by_kids
Tour de Ville	R037	45	26
Monstre	R026	42	26
Dragon	R010	40	30

Query 6

Objective: Find the 5 young customers who played the rides the most times in year 2019. This query helps La Ronde to find the young customers who had the greatest enthusiasm for all of the rides in the park.

Code:

```
SELECT COUNT(rt.ride_ticket_id) AS total_play_times, c.first_name, c.last_name
FROM rides_tickets AS rt
INNER JOIN Ticket AS t ON rt.ticket_ID = t.ticket_ID
INNER JOIN Customer AS c ON c.C_ID = t.customer_ID
WHERE t.ticket_id IN (
        SELECT DISTINCT t.ticket_id
        FROM ticket AS t
        INNER JOIN Customer AS c ON  t.customer_ID = c.C_ID
        WHERE c.age_interval = 'young')
AND YEAR(ride_access_time) = 2019
GROUP BY c.first_name
ORDER BY COUNT(ride_ticket_id) DESC
LIMIT 5;
```

Output:

total_play_times	first_name	last_name
31	Gregorio	Wetherill
18	Sibbie	Allwell
15	Sidonnie	Milroy
15	Dode	Koopman
15	Doreen	Barthelme

Query 7_____

Objective: Typically, dining facilities are not very popular when it is morning. Therefore, knowing and investing in the dining facilities that could attract customers even in the mornings is important. This query aims to find the dining facilities that were visited by more than 40 people in total in all of the mornings of 2020.

Code:

```
SELECT facility_desc, facility_type, facility_subtype, ft.count AS number_of_people_visited
FROM Facility AS f
INNER JOIN (
            SELECT facility_ID AS facid, COUNT(facility_ticket_ID) AS count
```

```
        FROM Facility_ticket
        WHERE YEAR(facility_access_time) = 2020
        AND HOUR(facility_access_time) < 12
        GROUP BY facility_ID
        HAVING COUNT(facility_ticket_ID) > 40
        ORDER BY COUNT(facility_ticket_ID) DESC
        ) AS ft ON ft.facid = f.facility_ID
WHERE facility_type = "Dinning";
```

Output:

facility_desc	facility_type	facility_subtype	number_of_people_visited
Fines Poutines Express	Dinning	Classic Food	53
Pizza Ronde 2	Dinning	Italian Food	52
Poulet Etc.	Dinning	Classic Food	47
Rolopan 1	Dinning	Ice Cream and Sweets	45
Bar Rouge	Dinning	Snacks & Beverages	45
Popcorn & Cie	Dinning	Snacks & Beverages	43
Queues De Castor 3	Dinning	Ice Cream and Sweets	42
Halte Gourmande - Au Bol	Dinning	Mediterranean Dishes	42
Popcorn & Cie	Dinning	Snacks & Beverages	42
Popcorn & Cie	Dinning	Snacks & Beverages	42
Popcorn & Cie	Dinning	Snacks & Beverages	42

Query 8_____

Objective: to find out the facilities of which the popularity is above average across all years.

Assumption: the "average of popularity" is an average value of popularity (visit times) of facilities across all years.

Code:

```
SELECT facility_desc, facility_type, facility_subtype, sub1.count1 AS visits
FROM Facility AS f
INNER JOIN (SELECT facility_ID AS facid1, COUNT(facility_ticket_ID) AS count1
        FROM Facility_ticket
        GROUP BY facility_ID) AS sub1 ON sub1.facid1 = f.facility_ID
WHERE sub1.count1 > (SELECT AVG(sub2.count) AS average_visit_times
        FROM (SELECT facility_ID AS facid, COUNT(facility_ticket_ID) AS count
                FROM Facility_ticket
                GROUP BY facility_ID) AS sub2 )
ORDER BY sub1.count1 DESC;
```

Output:

facility_desc	facility_type	facility_subtype	visits
Fines Poutines Express	Dinning	Classic Food	119
Popcorn & Cie	Dinning	Snacks & Beverages	114
Popcorn & Cie	Dinning	Snacks & Beverages	114
Carrousel Du Bonbon	Shopping	Candy	108
Au Comptoir Frais	Dinning	Light Dishes	107
Photo Goliath	Shopping	Photos and Collectables	106
Poulet Etc.	Dinning	Classic Food	105
Rolopan 1	Dinning	Ice Cream and Sweets	105
Pizza Ronde 2	Dinning	Italian Food	104
Boutique Splash	Shopping	Appareal	103
Moozoo 1	Dinning	Ice Cream and Sweets	102
Queues De Castor 3	Dinning	Ice Cream and Sweets	101
Popcorn & Cie	Dinning	Snacks & Beverages	101
La Centrale Burgers & Frites	Dinning	Classic Food	100
Pizza Ronde 1	Dinning	Italian Food	100

Query 9_____

Objective: La Ronde cares about customers' loyalty. This query is to find the customers whose times of visiting the park on different dates are above average (find the returning customers) in year 2020.

Sidenote: this query was constructed based on Rides_tickets, but removed duplicates that one person participated in multiple rides in his/her visit on one date by using "GROUP BY DAY(ride_access_time)".

Code:

```
SELECT CONCAT(c.first_name, " ", c.last_name) AS customer_name, COUNT(c.C_ID) AS visit_times_of_customer, c.C_ID
FROM Rides_tickets AS rt
INNER JOIN Ticket AS t ON rt.ticket_ID = t.ticket_ID
INNER JOIN Customer AS c ON c.C_ID = t.customer_ID
WHERE YEAR(ride_access_time) = 2020
GROUP BY c.C_ID, DAY(ride_access_time)
HAVING COUNT(c.C_ID) > (
    SELECT AVG(sub.visit_times_of_customer) AS avg_visit_times
    FROM (
        SELECT CONCAT(c.first_name, " ", c.last_name) AS name_c,
        COUNT(c.C_ID) AS visit_times_of_customer
        FROM Rides_tickets AS rt
        INNER JOIN Ticket AS t ON rt.ticket_ID = t.ticket_ID
        INNER JOIN Customer AS c ON c.C_ID = t.customer_ID
```

```
        WHERE YEAR(ride_access_time) = 2020
        GROUP BY c.C_ID, DAY(ride_access_time)) AS sub
        )
ORDER BY COUNT(c.C_ID) DESC;
```

Output:

customer_name	visit_times_of_customer	C_ID
Berta Drewery	11	CD0092
Janek Gerish	11	CD0031
Lianne Parkhouse	10	CD0148
Berta Drewery	10	CD0092
Jock Chasier	9	CD0035
Jae Mathivon	9	CD0039
Jeannine Lanchbery	9	CD0014
Rois Chrystie	9	CD0049
Roshelle Shah	9	CD0051
Rad Compford	9	CD0036
Colver Jira	9	CD0083
Jermain Meachem	8	CD0053
Dyane Kells	8	CD0135
Karol Gerbi	8	CD0008
Akim Preddle	8	CD0124

Query 10_____

Objective: from above query, we noticed that Berta Drewery (C_ID = “CD0092”), a loyal customer, visited La Ronde for 11 times in year 2020. This query is to find that, among the facilities of which the popularity is above average, which facility(ies) did Berta visited in year 2020.

Code:

```
SELECT CONCAT(c.first_name, " ", c.last_name) AS customer_name, f.facility_desc,
ft.facility_access_time
FROM Ticket AS t
INNER JOIN Customer AS c ON t.customer_ID = c.C_ID
INNER JOIN Facility_ticket AS ft ON ft.ticket_ID = t.ticket_ID
INNER JOIN Facility AS f ON f.facility_ID = ft.facility_ID
WHERE c.C_ID = "CD0092"
AND YEAR(facility_access_time) = 2020
AND facility_desc IN (
    SELECT f1.facility_desc
    FROM Facility AS f1
    INNER JOIN (SELECT facility_ID AS facid1, COUNT(facility_ticket_ID) AS count1
        FROM Facility_ticket
```

```
GROUP BY facility_ID) AS sub1 ON sub1.facid1 = f.facility_ID
WHERE sub1.count1 > (SELECT AVG(sub2.count) AS average_visit_times
                     FROM (SELECT facility_ID AS facid,
                                  COUNT(facility_ticket_ID) AS count
                           FROM Facility_ticket
                           GROUP BY facility_ID) AS sub2)
ORDER BY sub1.count1 DESC);
```

Output:

customer_name	facility_desc	facility_access_time
Berta Drewery	Rolopan 1	2020-01-30 22:40:00
Berta Drewery	La Centrale Burgers & Frites	2020-01-30 23:55:00
Berta Drewery	Bar Rouge	2020-06-18 11:15:00
Berta Drewery	Popcorn & Cie	2020-04-30 15:19:00
Berta Drewery	Poulet Etc.	2020-04-30 13:22:00
Berta Drewery	Popcorn & Cie	2020-04-06 12:05:00
Berta Drewery	Carrousel Du Bonbon	2020-04-06 01:33:00
Berta Drewery	Halte Gourmande - Le Ma...	2020-04-30 19:44:00
Berta Drewery	Subway	2020-04-30 12:21:00
Berta Drewery	La Centrale Burgers & Frites	2020-04-30 01:42:00
Berta Drewery	Carrousel Du Bonbon	2020-04-30 18:43:00
Berta Drewery	Photo Goliath	2020-04-30 18:49:00
Berta Drewery	Popcorn & Cie	2020-03-17 15:16:00
Berta Drewery	Popcorn & Cie	2020-03-17 16:31:00
Berta Drewery	Au Comptoir Frais	2020-03-17 02:15:00