INSY 661-075

Data and Distributed Systems for Analytics

Presented to Professor Animesh Animesh

Individual Project

Prepared by Cristina Esposito 260744222

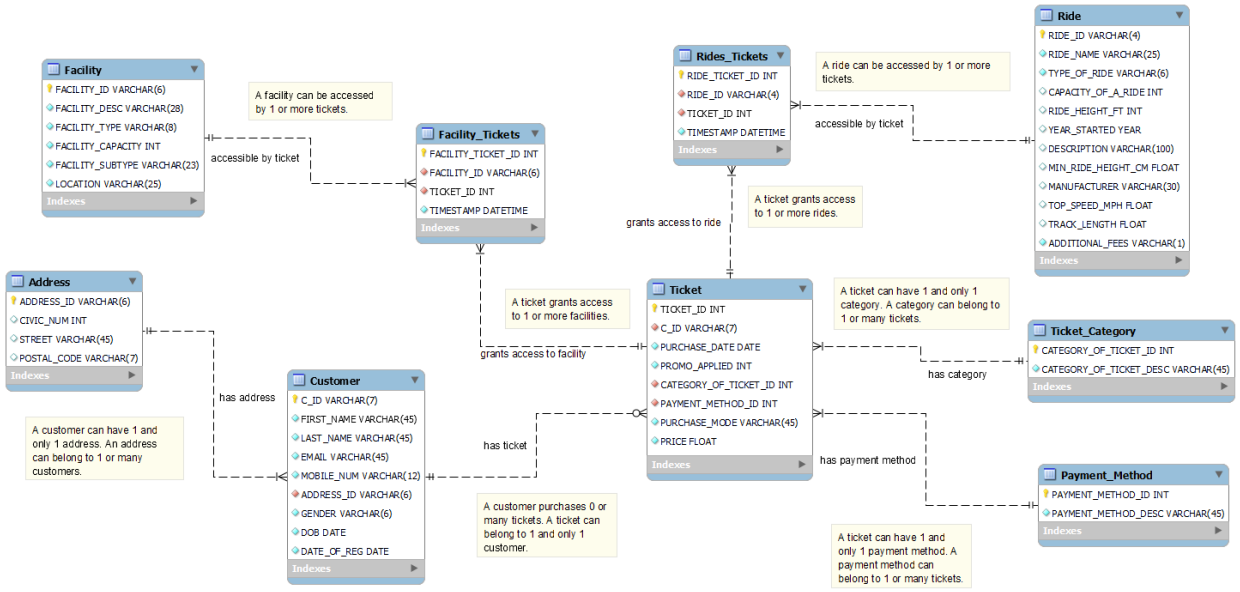
Friday, September 10th, 2021

Contents

PART 1 – Original La Ronde Database 1
1.0 ERD 1
2.0 Relational Model 2
3.0 Queries 2
3.1 Query 1 2
3.2 Query 2 3
3.3 Query 3 5
3.4 Query 4 6
3.5 Query 5 7
3.6 Query 6 8
3.7 Query 7 9
3.8 Query 8 10
3.9 Query 9 12
3.10 Query 10 14
PART 2 – Modified La Ronde Database 16
4.0 Modified ERD 16
5.0 Modified Relational Model 17
6.0 Queries 17
6.1 Query 1 17
6.2 Query 2 18
6.3 Query 3 19
6.4 Query 4 20
6.5 Query 5 21
6.6 Query 6 (Bonus Query) 21
Appendix 23

PART 1 – Original La Ronde Database

1.0 ERD



Assumptions: This ERD was made in MySQL Workbench. When adding the relationships between entities, it automatically adds the FK. It also automatically creates bridge tables for many-to-many relationships.

Data Anomalies:

- A separate table to store all the addresses was created since the address is multi-valued
- A payment method table was created to store the payment method information that is on a ticket
- A ticket category table was created to store the ticket category information
- The facility table needed to be cleaned to have only the unique list of facilities and the attributes of those facilities
- A facility tickets table was created with the records from the original facility excel file but only keeping the facility ID, ticket ID, and timestamp, and added a unique identifier called facility ticket ID
- Dates on the excel files needed to be manipulated to be in YYYY-MM-DD format for MySQL to recognize the date format
- Some customer's DOB comes after their registration date. This was ignored and no manipulation was done to the data.
- The Customer ID "CD00010" in the ticket table is not in the customer's table. I decided to keep this ID the way it is and not assume its information should be for "CD0010" or "CD0001". In order to overcome the error messages because of the "child" not belonging to the "parent", the following code was used when creating the tables and inserting the data:

```
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

2.0 Relational Model

Address (ADDRESS_ID, CIVIC_NUM, STREET, POSTAL_CODE)

Customer (C_ID, FIRST_NAME, LAST_NAME, EMAIL, MOBILE_NUM, ADDRESS_ID, GENDER, DOB, DATE_OF_REG)

Ticket (TICKET_ID, C_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)

Facility_Tickets (FACILITY_TICKET_ID, FACILITY_ID, TICKET_ID, TIMESTAMP)

Rides_Tickets (RIDE_TICKET_ID, RIDE_ID, TICKET_ID, TIMESTAMP)

Ride (RIDE_ID, RIDE_NAME, TYPE_OF_RIDE, CAPACITY_OF_RIDE, RIDE_HEIGHT_FT, YEAR_STARTEDM DESCRIPTION, MIN_RIDE_HEIGHT_CM, MANUFACTURER, TOP_SPEED_MPH, TRACK_LENGTH, ADDITIONAL_FEES)

Ticket_Category (CATEGORY_OF_TICKET_ID, CATEGORY_OF_TICKET_DESC)

Payment_Method (PAYMENT_METHOD_ID, PAYMENT_METHOD_DESC)

3.0 Queries

3.1 Query 1

Objective: Find out how much revenue was lost due to promotions applied on tickets.

Assumptions: Total revenue is considered as the sum of the prices paid on each ticket by customer. Loss in revenue is calculated by the difference between the price paid and the regular full price, divided by the regular full price.

Code:

```
/* Step 1 - Create a view with the pricing options by category and promotion applied*/
create view pricing_options as
SELECT ticket.PROMO_APPLIED,ticket.CATEGORY_OF_TICKET_ID,
ticket_category.CATEGORY_OF_TICKET_DESC, ticket.price
FROM ticket, ticket_category
where ticket.CATEGORY_OF_TICKET_ID=ticket_category.CATEGORY_OF_TICKET_ID
group by PROMO_APPLIED, CATEGORY_OF_TICKET_ID;

/*Step 2 - Create a view table that has the regular full price of what the person should have paid if they
purchased with a promotion*/
create view ticketsfullprice as
select *,
case
        when PROMO_APPLIED=1 and CATEGORY_OF_TICKET_ID=1 then 800
```

```
        when PROMO_APPLIED=1 and CATEGORY_OF_TICKET_ID=3 then 150
        else price
end as RegularFullPrice
from ticket;

/*Step 3
    - Sum the difference between full price and price paid to get the revenue lost from promotions
    - Divide the different of price paid and regular price by regular price to get the equivalent
      percentage of revenue loss*/
select concat("$",sum(RegularFullPrice-PRICE)) as "Total Loss in Revenue From Promotion",
concat(format((sum(PRICE-RegularFullPrice)/sum(RegularFullPrice)*100),2),"%") as "Percentage Loss
in Revenue"
from ticketsfullprice;
```

Output:
From the view table "pricing_options":

PROMO_APPLIED	CATEGORY_OF_TICKET_ID	CATEGORY_OF_TICKET_DESC	price
1	1	Annual pass	700
0	1	Annual pass	800
1	2	Parking ticket	20
0	2	Parking ticket	20
1	3	Daily Pass	120
0	3	Daily Pass	150

From the view table "ticketsfullprice":

TICKET_ID	C_ID	PURCHASE_DATE	PROMO_APPLIED	CATEGORY_OF_TICKET_ID	PAYMENT_METHOD_ID	PURCHASE_MODE	PRICE	RegularFullPrice
1	CD0143	2020-06-27	1	1	2	Online	700	800
2	CD0109	2019-11-27	1	1	1	Online	700	800
3	CD0149	2020-04-18	1	3	2	Online	120	150
4	CD0053	2020-07-24	1	1	4	Online	700	800
5	CD0038	2019-12-18	1	1	2	Online	700	800
6	CD0019	2019-12-03	1	1	2	Offline	700	800

From the final output:

Total Loss in Revenue From Promotion	Percentage Loss in Revenue
$22950	-6.91%

Interpretation: La Ronde has lost $22,950 in revenue from promotions, which equates to 6.91% loss in ticket sales revenue from promotions.

3.2 Query 2

Objective: To see which type of ride has been used the most by each age group. We want to see this breakdown by age group to understand which types of rides they prefer/don't prefer to help La Ronde identify areas of improvement.

Assumptions: A customer was put into an age group based on Statistics Canada's definition of each age group (https://www.statcan.gc.ca/eng/concepts/definitions/previous/age1a). Each type of ride could have rides that a customer has frequented more than once. This is to help us see the true usage of each type of ride.

Code:

```
/* Step 1 - adding new columns "Age", "Years registered", and "Age groups" and creating it as a new customer view table*/
create view customers_age as
select*, timestampdiff(year, DOB, now()) as Age, timestampdiff(year, DATE_OF_REG, now()) as YearsRegistered,
case
        when timestampdiff(year, DOB, now())<=14 then "Child"
        when timestampdiff(year, DOB, now())<=24 then "Youth"
        When timestampdiff(year, DOB, now())<=64 then "Adult"
else "Senior"
end as AgeGroup
from customer;

/* Step 2 - See how many times a type of ride has been used by each age group. */
select ride.TYPE_OF_RIDE, customers_age.Agegroup, count(customers_age.Agegroup) as TotalAgeGroup, concat(format((count(customers_age.Agegroup)/(select count(*) from ride, rides_tickets, ticket, customers_age
where ride.RIDE_ID=rides_tickets.RIDE_ID and rides_tickets.TICKET_ID=ticket.TICKET_ID and ticket.C_ID=customers_age.C_ID)*100),2),"%") as PercentageTotalAgeGroup
from ride, rides_tickets, ticket, customers_age
where ride.RIDE_ID=rides_tickets.RIDE_ID and rides_tickets.TICKET_ID=ticket.TICKET_ID and ticket.C_ID=customers_age.C_ID
group by ride.TYPE_OF_RIDE, Agegroup
order by ride.TYPE_OF_RIDE, (count(customers_age.Agegroup)/(select count(*) from ride, rides_tickets, ticket, customers_age
where ride.RIDE_ID=rides_tickets.RIDE_ID and rides_tickets.TICKET_ID=ticket.TICKET_ID and ticket.C_ID=customers_age.C_ID)) desc;
```

Output:

From the view table “customers_age”

C_ID	FIRST_NAME	LAST_NAME	EMAIL	MOBILE_NUM	ADDRESS_ID	GENDER	DOB	DATE_OF_REG	Age	YearsRegistered	AgeGroup
CD0001	Maitilde	Arnholtz	marnholtz0@census.gov	781-437-0202	AD0001	Female	1968-11-18	2020-04-12	52	1	Adult
CD0002	Tamarra	Bridewell	tbridewell1@tamu.edu	131-119-5300	AD0002	Female	2002-02-18	2020-04-25	19	1	Youth
CD0003	Aggi	Fearey	afearey2@stanford.edu	365-718-2859	AD0003	Female	2002-06-30	2020-06-30	19	1	Youth
CD0004	Jaimie	Hanster	jhanster3@cnbc.com	321-723-9654	AD0004	Male	1969-05-04	2018-05-12	52	3	Adult
CD0005	Andres	Newcomb	anewcomb4@unesco.org	511-439-1039	AD0005	Male	1982-06-29	2017-04-19	39	4	Adult
CD0006	Vinnie	Brownhill	vbrownhill5@godaddy.com	410-447-2896	AD0006	Male	2013-04-06	2018-03-03	8	3	Child
CD0007	Yelena	Triggs	ytriggs6@wikia.com	861-823-8373	AD0007	Female	2016-10-30	2019-01-11	4	2	Child

From the final output:

TYPE_OF_RIDE	AgeGroup	TotalAgeGroup	PercentageTotalAgeGroup
Family	Adult	852	21.39%
Family	Child	317	7.96%
Family	Youth	288	7.23%
Kids	Adult	565	14.19%
Kids	Child	249	6.25%
Kids	Youth	188	4.72%
Thrill	Adult	875	21.97%
Thrill	Child	326	8.18%
Thrill	Youth	323	8.11%

Interpretation: Adults make up the biggest percentage of users per type of ride. Interestingly, we can see that all age groups prefer thrill rides (21.97% adults, 8.18% children, and 8.11% youth), followed by family rides (21.39% adults, 7.96% children, and 7.23% youth, and lastly kids rides (14.19% adults, 6.25% children, and 4.72% youth). If La Ronde wants to attract more customer and keep them in the park longer, they can look at removing kids rides and adding new thrill rides.

3.3 Query 3

Objective: To understand the relationship between the type of facility and its location near rides. This can help La Ronde understand where to best place their facilities near certain areas of the park.

Assumptions: A facility's usage is based on the number of times it is recorded in facility_tickets. A facility is located near a ride when the "Location" states "near the" or "next to the".

Code:

```
create view FacilityRideProximity as
select *,
case
        when location like '%Near%' then "Near Ride"
        when location like '%Next%' then "Near Ride"
        else "Not near a ride"
end as RideProximity
from facility;

/* Step 2 - Seeing which type of facilities perform better in which locations*/
select FacilityRideProximity.FACILITY_TYPE, FacilityRideProximity.RideProximity,
count(FacilityRideProximity.FACILITY_ID) as "Facility Usage"
from FacilityRideProximity, facility_tickets
where FacilityRideProximity.FACILITY_ID=facility_tickets.FACILITY_ID
group by FacilityRideProximity.FACILITY_TYPE, FacilityRideProximity.RideProximity
order by count(FacilityRideProximity.FACILITY_ID) desc;
```

Output:

From the view table "FacilityRideProximity"

FACILITY_ID	FACILITY_DESC	FACILITY_TYPE	FACILITY_CAPACITY	FACILITY_SUBTYPE	LOCATION	RideProximity
FAC101	Emporium	Shopping	20	Toys/Appareal/Sundries	Front entrance	Not near a ride
FAC102	Marchand Du Village	Shopping	20	Toys/Appareal/Sundries	Secteur Village	Not near a ride
FAC103	Photo Goliath	Shopping	20	Photos and Collectables	Near the Goliath	Near Ride
FAC104	Carrousel Du Bonbon	Shopping	20	Candy	Front entrance	Not near a ride
FAC105	ChÃ¢teau Du Bonbon	Shopping	20	Candy	Secteur Village	Not near a ride
FAC106	Boutique Du Far West	Shopping	20	Appareal	Fort Edmonton	Not near a ride
FAC107	Boutique Spirale	Shopping	20	Appareal	Next to Spirale	Near Ride
FAC108	Boutique Splash	Shopping	20	Appareal	Next to Splash	Near Ride
FAC109	Yo! Chine	Dinning	20	Asian Food	Next to Splash	Near Ride
FAC110	Fines Poutines Express	Dinning	20	Classic Food	Next to Splash	Near Ride

From the final output:

	FACILITY_TYPE	RideProximity	Facility Usage
▶	Dinning	Near Ride	2064
	Dinning	Not near a ride	1112
	Shopping	Not near a ride	441
	Shopping	Near Ride	297
	Events	Not near a ride	85

Interpretation: Dining facilities tend to see the highest usage when they are located near a ride, and shopping facilities tend to see higher usage when they are not near a ride. La Ronde should locate these types of facilities accordingly with their location near a ride.

3.4 Query 4

Objective: Understand the behaviour of customers when they come to the park regarding where they start and where they finish.

Assumptions: If a customer is only registered at one location, we can assume that the location is the first and last place they went to. The last ride a customer took may be from a completely different day than the date from their first ride.

Code:

```
/* Step 1 - Combine the two tables together using a union*/
create view facilityridecombined as
select rides_tickets.RIDE_ID as FacilityRideID, "Ride" as FaciltyorRide, rides_tickets.TICKET_ID,
rides_tickets.timestamp from rides_tickets
union
select facility_tickets.FACILITY_ID, "Facility" as FaciltyorRide, facility_tickets.TICKET_ID,
facility_tickets.timestamp from facility_tickets
order by TICKET_ID,timestamp;

/*Step 2 - Identify the first and last place the customer attended*/
create view firstandlast as
select Tmin.TICKET_ID, Tmin.FacilityRideID as StartPlaceID, Tmin.FaciltyorRide as StartPlace,
Tmin.timestamp StartDate, Tmax.FacilityRideID as EndPlaceID, Tmax.FaciltyorRide as EndPlace,
Tmax.timestamp as EndDate
from (select* from facilityridecombined
where timestamp= (select min(timestamp) from facilityridecombined as f2 where
f2.TICKET_ID=facilityridecombined.TICKET_ID)) as Tmin,
(select *
from facilityridecombined
where timestamp= (select max(timestamp) from facilityridecombined as f2 where
f2.TICKET_ID=facilityridecombined.TICKET_ID)) as Tmax
where Tmin.TICKET_ID=Tmax.TICKET_ID;

/*Step 3 - See the pattern of customers*/
select StartPlace, EndPlace, count(*) as NumCustomers
from firstandlast
```

group by StartPlace, EndPlace
order by NumCustomers desc;

Output:
From the view table “facilityridecombined”:

FacilityRideID	FaciltyorRide	TICKET_ID	timestamp
R020	Ride	1	2020-06-26 22:10:00
R028	Ride	1	2020-06-26 22:58:00
R024	Ride	1	2020-06-27 07:20:00
R027	Ride	1	2020-06-27 12:43:00
FAC112	Facility	1	2020-06-27 23:47:00
R017	Ride	2	2019-11-26 16:50:00
R027	Ride	2	2019-11-26 23:10:00
FAC113	Facility	2	2019-11-27 13:52:00
FAC136	Facility	2	2019-11-27 18:28:00
FAC139	Facility	2	2019-11-27 20:33:00
FAC107	Facility	2	2019-11-27 23:06:00
FAC127	Facility	2	2019-11-27 23:53:00

From the view table “firstandlast”:

TICKET_ID	StartPlaceID	StartPlace	StartDate	EndPlaceID	EndPlace	EndDate
1	R020	Ride	2020-06-26 22:10:00	FAC112	Facility	2020-06-27 23:47:00
2	R017	Ride	2019-11-26 16:50:00	FAC127	Facility	2019-11-27 23:53:00
3	R034	Ride	2020-04-17 21:15:00	FAC110	Facility	2020-04-18 23:57:00
4	R003	Ride	2020-07-23 15:46:00	FAC111	Facility	2020-07-24 21:10:00
5	R009	Ride	2019-12-17 20:20:00	FAC141	Facility	2019-12-18 10:37:00
6	R010	Ride	2019-12-02 14:09:00	FAC125	Facility	2019-12-03 17:02:00
7	R003	Ride	2020-06-17 02:54:00	FAC141	Facility	2020-06-17 15:33:00
8	R035	Ride	2020-02-01 21:56:00	FAC141	Facility	2020-02-02 21:12:00
9	R011	Ride	2020-02-05 14:29:00	FAC121	Facility	2020-02-06 10:34:00
10	R020	Ride	2019-11-28 18:11:00	FAC134	Facility	2019-11-29 17:07:00

From the final output:

StartPlace	EndPlace	NumCustomers
Ride	Facility	805
Facility	Facility	98
Ride	Ride	90
Facility	Ride	9

Interpretation: We can see that the vast majority of customers that come to the park start with a ride and end with a facility. This could be that customers want to go on rides first when they arrive and finish their day at a facility. This could help La Ronde configure their park in such a way where they put many of their rides further into the park to maximize the customer’s time in the park, and then put facilities towards the entrance/exit of the park as they are on their way out.

3.5 Query 5

Objective: To see if there is a relationship between the category of ticket, purchase mode, and the month purchased. This can help La Ronde plan for in-person staffing and potentially set up certain ticket booths to service the type of ticket.

Code:

```
select case
        when month(PURCHASE_DATE)=1 then "January"
        when month(PURCHASE_DATE)=2 then "February"
        when month(PURCHASE_DATE)=3 then "March"
        when month(PURCHASE_DATE)=4 then "April"
        when month(PURCHASE_DATE)=5 then "May"
        when month(PURCHASE_DATE)=6 then "June"
        when month(PURCHASE_DATE)=7 then "July"
        when month(PURCHASE_DATE)=8 then "August"
        when month(PURCHASE_DATE)=9 then "September"
        when month(PURCHASE_DATE)=10 then "October"
        when month(PURCHASE_DATE)=11 then "November"
        else "December"
end as MonthPurchased, ticket_category.CATEGORY_OF_TICKET_DESC,
ticket.PURCHASE_MODE, count(*) as NumPurchases
from ticket, ticket_category
where ticket.CATEGORY_OF_TICKET_ID=ticket_category.CATEGORY_OF_TICKET_ID
group by MonthPurchased, CATEGORY_OF_TICKET_DESC, PURCHASE_MODE
order by NumPurchases desc limit 10;
```

Output:

MonthPurchased	CATEGORY_OF_TICKET_DESC	PURCHASE_MODE	NumPurchases
January	Parking ticket	Offline	30
December	Annual pass	Offline	26
April	Daily Pass	Online	25
April	Annual pass	Offline	25
December	Daily Pass	Offline	24
March	Daily Pass	Online	24
May	Annual pass	Online	24
December	Parking ticket	Offline	22
July	Annual pass	Online	21
June	Annual pass	Offline	21

Interpretation: La Ronde may want to staff more parking attendants in January and December since most parking passes were purchased in person during these two months. La Ronde may also want to staff more people at the ticket booth in December, as most customers purchased daily and annual passes in person during this month.

3.6 Query 6

Objective: Related with the previous query (3.5), we want to see if there is a relationship between the category of ticket, purchase mode, and the day of the week the ticket was purchased. This can help La Ronde plan for in-person staffing and potentially set up certain ticket booths to service the type of ticket.

Code:

```
select
case
        when weekday(PURCHASE_DATE)=1 then "Tuesday"
        when weekday(PURCHASE_DATE)=2 then "Wednesday"
```

```
        when weekday(PURCHASE_DATE)=3 then "Thursday"
        when weekday(PURCHASE_DATE)=4 then "Friday"
        when weekday(PURCHASE_DATE)=5 then "Saturday"
        when weekday(PURCHASE_DATE)=6 then "Sunday"
        else "Monday"
end as Weekday, ticket_category.CATEGORY_OF_TICKET_DESC, ticket.PURCHASE_MODE,
count(*) as NumPurchases
from ticket, ticket_category
where ticket.CATEGORY_OF_TICKET_ID=ticket_category.CATEGORY_OF_TICKET_ID
group by Weekday, CATEGORY_OF_TICKET_DESC, PURCHASE_MODE
order by NumPurchases desc limit 10;
```

Output:

Weekday	CATEGORY_OF_TICKET_DESC	PURCHASE_MODE	NumPurchases
Wednesday	Daily Pass	Online	38
Thursday	Daily Pass	Offline	32
Saturday	Annual pass	Online	29
Sunday	Annual pass	Online	29
Saturday	Parking ticket	Offline	29
Thursday	Annual pass	Offline	29
Wednesday	Annual pass	Online	28
Sunday	Parking ticket	Offline	28
Monday	Daily Pass	Offline	28
Monday	Annual pass	Offline	28

Interpretation: Daily passes are purchased the most on Thursdays in person, annual passes tend to be purchased most on Sundays in person, and ticket passes tend to be purchased most on Saturdays in person. La Ronde may want to staff more people at ticket booths on these days.

3.7 Query 7

Objective: To see which rides are used the most and at which times to ensure that maintenance and safety checks are performed more frequently, and further avoid future breakdowns.

Assumptions: The classification of time of day was taken from the following website: https://wgntv.com/weather/how-do-you-define-daytime-and-evening-times-in-a-weather-forecast/

Code:

```
/* Step 1 - Create a view with the time of day each ride was used at*/
create view rideusedtimeofday as
select RIDE_NAME,
case
        when time(timestamp)<'06:00:00' then "Late at night"
        when time(timestamp)<'09:00:00' then "Early morning"
        when time(timestamp)<'12:00:00' then "Late morning"
        when time(timestamp)<'15:00:00' then "Early Afternoon"
        when time(timestamp)<'18:00:00' then "Late afternoon"
        else "Late evening"
end as TimeofDay, count(*) as numused
from rides_tickets, ride
```

```
where ride.RIDE_ID=rides_tickets.RIDE_ID
group by RIDE_NAME, TimeofDay
order by RIDE_NAME, numused desc;

/* Step 2 - See at which time of day is the most popularly used on each ride*/
select TimeofDay, count(TimeofDay) as "Number of rides"
from (select ride_name, TimeofDay, max(numused)
        from rideusedtimeofday
        group by ride_name) as maxtimeofday
group by timeofday;
```

Output:
From the view table “rideusedtimeofday”:

RIDE_NAME	TimeofDay	numused
Air Papillon	Late evening	30
Air Papillon	Late at night	18
Air Papillon	Late afternoon	16
Air Papillon	Late morning	13
Air Papillon	Early morning	10
Air Papillon	Early Afternoon	7
Aqua Twist	Late evening	30
Aqua Twist	Late at night	24
Aqua Twist	Late morning	12
Aqua Twist	Early Afternoon	11
Aqua Twist	Late afternoon	10
Aqua Twist	Early morning	10
Autos Tam…	Late evening	27
Autos Tam…	Late at night	25

From the final output:

TimeofDay	Number of rides
Late evening	31
Late at night	11

Interpretation: 31 rides are mostly used late in the evening and 11 rides are used mostly late at night. La Ronde may want to do maintenance servicing and safety checks not during these times to maximize customers usage of the rides.

3.8 Query 8

Objective: To see if the height limits on rides are being respected. This will help La Ronde prevent future injuries, especially amongst children, and avoid getting fined by inspectors for not respecting safety protocols.

Assumptions: We can assume that the average height of each age group and gender for children and teenagers is according to this table from Proxim, a Quebec Pharmacy: (https://www.groupeproxim.ca/en/article/health/average-height-and-weight). The analysis will only look at children under the age of 10, since at age 10 they meet the average height of all the rides that have height requirements.

Code:

```
/* Step 1 - Create a view table with the heights per gender and age. We can get the age
from the view table already done in query 2 "customers_age"*/
create view rideVScustomerHeight as
SELECT RIDE_NAME, MIN_RIDE_HEIGHT_CM, customers_age.C_ID, Gender, dob, age,
case
        when Age<=1 and gender="Female" then 74
        when Age<=1 and gender="Male" then 76
        when Age<=2 and gender="Female" then 87
        when Age<=2 and gender="Male" then 88
        when Age<=3 and gender="Female" then 95
        when Age<=3 and gender="Male" then 96
        when Age<=4 then 103
        when Age<=5 and gender="Female" then 109
        when Age<=5 and gender="Male" then 110
        when Age<=6 and gender="Female" then 115
        when Age<=6 and gender="Male" then 116
        when Age<=7 and gender="Female" then 121
        when Age<=7 and gender="Male" then 122
        when Age<=8 then 127
else 133
end as HeightCM
FROM ride, rides_tickets, ticket, customers_age
where MIN_RIDE_HEIGHT_CM is not null and ride.RIDE_ID=rides_tickets.RIDE_ID and
rides_tickets.TICKET_ID=ticket.TICKET_ID and customers_age.C_ID=ticket.C_ID
having MIN_RIDE_HEIGHT_CM>HeightCM;

/* Step 2 - See the total number of times the height limit was not respected*/
SELECT count(*) as "Number of Infractions"
FROM rideVScustomerHeight;

/* Step 3 - See which rides are the "repeat offenders" */
SELECT RIDE_NAME, count(*) as NumberOfInfractions, min(age) as YougestAge
FROM rideVScustomerHeight
where MIN_RIDE_HEIGHT_CM>HeightCM
group by RIDE_NAME
order by NumberOfInfractionsdesc;
```

Output:

From the view table “rideVScustomerHeight”:

RIDE_NAME	MIN_RIDE_HEIGHT_CM	C_ID	Gender	dob	age	HeightCM
La Grande ...	91.4	CD0009	Female	2020-05-01	1	74
Ourson Fri...	91.4	CD0009	Female	2020-05-01	1	74
La Grande ...	91.4	CD0009	Female	2020-05-01	1	74
Ourson Fri...	91.4	CD0089	Male	2019-03-29	2	88
La Marche ...	91.4	CD0089	Male	2019-03-29	2	88
Aqua Twist	91.4	CD0089	Male	2019-03-29	2	88
Ourson Fri...	91.4	CD0089	Male	2019-03-29	2	88
Toboggan ...	106.7	CD0007	Female	2016-10-30	4	103
Splash	106.7	CD0007	Female	2016-10-30	4	103
Toboggan ...	106.7	CD0007	Female	2016-10-30	4	103
Toboggan ...	106.7	CD0009	Female	2020-05-01	1	74
Dragon	106.7	CD0009	Female	2020-05-01	1	74

From the output with the total number of infractions:

	Number of Infractions
▶	439

From the output with the total number of infractions by ride:

	RIDE_NAME	NumberOfInfractions	YougestAge
▶	Goliath	96	1
	Vampire	88	4
	DÃ©mon	70	3
	EdnÃ¶r	16	1
	VipÃ¨re	15	1
	Boomerang	11	2
	Catapulte	11	2
	Manitou	11	3
	Condor	10	3
	Disco Ronde	10	1
	Monstre	10	1
	Orbite	10	3
	Tour de Ville	10	3

Interpretation: La Ronde has had a total of 439 infractions of not respecting the height limits on their own rides. The Goliath, Vampire, and Démon have the highest number of infractions, and many of these rides have had customers as young as 1 years old on them. La Ronde should be stricter in enforcing their height limits if they do not want to be fined or shut down by safety inspectors.

3.9 Query 9

Objective: Recommender system for rides. We want to see which customer has been on the most rides (CD0035) and find the customer ID that has also been on the same rides the most.

Assumptions: We want to see only the distinct number of times someone has been on a ride.

Code:

```
/* Step 1 - Identify the customer that has been on the most rides, customer CD0035*/
select C_ID, count(distinct ride_id) as NumberRides
from rides_tickets, ticket
where rides_tickets.ticket_id=ticket.TICKET_ID
group by C_ID
order by NumberRides desc;

/* Step 2 - Find all the unique rides customer CD0035 has been on*/
create view CD0035Rides as
select C_ID, ride_id
from rides_tickets, ticket
where rides_tickets.ticket_id=ticket.TICKET_ID and C_ID="CD0035"
group by ride_id;
```

```
/* Step 3 - Do a cartesian to see all the customers that have also been on the same rides as CD0035*/
create view cartesianCD0035 as
select *
from cd0035rides, (select C_ID as allc_id, ride_id as allridesid
                   from rides_tickets, ticket
                   where rides_tickets.ticket_id=ticket.TICKET_ID) as allcustrides
where ride_id=allridesid
group by allc_id, allridesid;
/* Step 4 - See which customers are the most similar to CD0035*/
select allc_id, count(distinct allridesid) as numsimilar
from cartesiancd0035
where allc_id not in ("CD0035")
group by allc_id
order by numsimilar desc limit 2;
```

Output:
From the first query to see which customer has been on the most rides:

C_ID	NumberRides
CD0035	34
CD0150	33
CD0022	31
CD0124	31
CD0039	30
CD0092	30
CD0093	30
CD0144	30

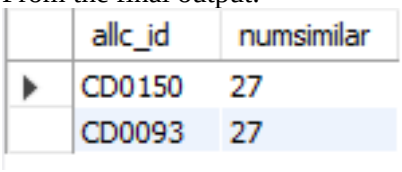
From the view table “CD0035Rides”:

C_ID	ride_id
CD0035	R005
CD0035	R025
CD0035	R003
CD0035	R021
CD0035	R007
CD0035	R013
CD0035	R001
CD0035	R002

From the view table “cartesianCD0035”:

C_ID	ride_id	allc_id	allridesid
CD0035	R005	CD0149	R005
CD0035	R005	CD0090	R005
CD0035	R005	CD0144	R005
CD0035	R005	CD0089	R005
CD0035	R005	CD0146	R005
CD0035	R005	CD0114	R005

From the final output:



	allc_id	numsimilar
▶	CD0150	27
	CD0093	27

Interpretation: There are two customers that have been on the same number of rides as CD0035: CD0150 and CD0093. La Ronde can look into building an app where it can allow these two customers to put in the 27 rides they have been on and provide suggestions of other rides that CD0035 has been on.

3.10 Query 10

Objective: Recommender system for facilities. Want to see which customer has been to the most facilities (CD0083) and find the customer ID that has also been to the same facilities the most.

Assumptions: We want to see only the distinct number of times someone has been to a facility.

Code:

```
/* Step 1 - Identify the customer that has been to the most facilities, customer CD0083*/
select C_ID, count(distinct FACILITY_ID) as NumberFacilities
from facility_tickets, ticket
where facility_tickets.ticket_id=ticket.TICKET_ID
group by C_ID
order by NumberFacilities desc;

/* Step 2 - Find all the unique facilities customer CD0083 has been to*/
create view CD0083Facilities as
select C_ID, FACILITY_ID
from facility_tickets, ticket
where facility_tickets.ticket_id=ticket.TICKET_ID and C_ID="CD0083"
group by FACILITY_ID;

/* Step 3 - Do a cartesian to see all the customers that have also been to the same facilities as CD0083*/
create view cartesianCD0083 as
select *
from CD0083Facilities, (select C_ID as allc_id, FACILITY_ID as allfacilitiesid
                        from facility_tickets, ticket
                        where facility_tickets.ticket_id=ticket.TICKET_ID) as allcustfacilities
where FACILITY_ID=allfacilitiesid
group by allc_id, allfacilitiesid;

/* Step 4 - See which customers are the most similar to CD0083*/
select allc_id, count(distinct allfacilitiesid) as numsimilar
from cartesianCD0083
where allc_id not in ("CD0083")
group by allc_id
order by numsimilar desc limit 2;
```

Output:
From the first query to see which customer has been to the most facilities:

C_ID	NumberFacilities
CD0083	32
CD0005	31
CD0022	31
CD0053	31
CD0087	31
CD0093	31
CD0148	31

From the view table "CD0083Facilities":

C_ID	FACILITY_ID
CD0083	FAC102
CD0083	FAC104
CD0083	FAC107
CD0083	FAC133
CD0083	FAC134
CD0083	FAC103
CD0083	FAC105

From the view table "cartesiancd0083":

C_ID	FACILITY_ID	allc_id	allfacilitiesid
CD0083	FAC102	CD0053	FAC102
CD0083	FAC102	CD0133	FAC102
CD0083	FAC102	CD0142	FAC102
CD0083	FAC102	CD0023	FAC102
CD0083	FAC102	CD0107	FAC102
CD0083	FAC102	CD0052	FAC102
CD0083	FAC102	CD0048	FAC102
CD0083	FAC102	CD0066	FAC102
CD0083	FAC102	CD0145	FAC102

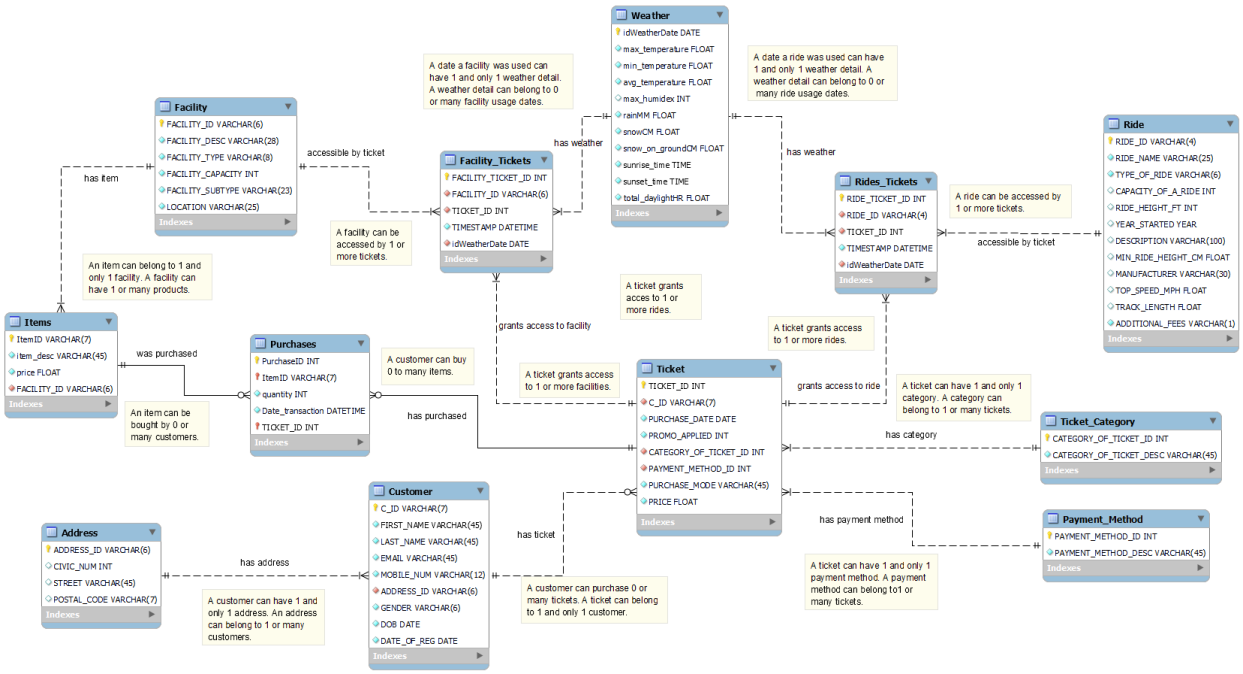
From the final output:

allc_id	numsimilar
CD0005	24
CD0022	24

Interpretation: There are two customers that have been to the same number of facilities as CD0083: CD005 and CD0022. Like query 10, La Ronde can build an app where it can allow these two customers to put in the 24 facilities they have been to and provide suggestions of other facilities that CD0083 has been to.

PART 2 – Modified La Ronde Database

4.0 Modified ERD



Assumptions: This ERD was made in MySQL Workbench. When adding the relationships between entities, it automatically adds the FK. It also automatically creates bridge tables for many-to-many relationships.

Modifications:

- The modified database incorporates a weather table for Montreal, an items table with all the items customers can buy from facilities, and a purchases table to keep track of all the purchases done by a customer when they visited a facility.
- The weather data for Montreal was extracted from this website: https://montreal.weatherstats.ca/download.html
- The items table was inspired by some of the following websites for prices and product offerings:
  - https://www.restoamir.com/english/menu
  - https://shop.dippindots.com/cotton-candy/
  - https://beavertails.com/products/
- Most of the information in the items and purchases table are fake and are used for demonstration purposes for why it would be beneficial for La Ronde to store this information and see the value from it.

5.0 Modified Relational Model

Address (ADDRESS_ID, CIVIC_NUM, STREET, POSTAL_CODE)

Customer (C_ID, FIRST_NAME, LAST_NAME, EMAIL, MOBILE_NUM, ADDRESS_ID, GENDER, DOB, DATE_OF_REG)

Ticket (TICKET_ID, C_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)

Facility_Tickets (FACILITY_TICKET_ID, FACILITY_ID, TICKET_ID, TIMESTAMP, idWeatherDate)

Rides_Tickets (RIDE_TICKET_ID, RIDE_ID, TICKET_ID, TIMESTAMP, idWeatherDate)

Ride (RIDE_ID, RIDE_NAME, TYPE_OF_RIDE, CAPACITY_OF_RIDE, RIDE_HEIGHT_FT, YEAR_STARTEDM DESCRIPTION, MIN_RIDE_HEIGHT_CM, MANUFACTURER, TOP_SPEED_MPH, TRACK_LENGTH, ADDITIONAL_FEES)

Ticket_Category (CATEGORY_OF_TICKET_ID, CATEGORY_OF_TICKET_DESC)

Payment_Method (PAYMENT_METHOD_ID, PAYMENT_METHOD_DESC)

Items (ItemID, item_desc, price, FACILITY_ID)

Purchases (PurchaseID, Item_ID, quantity, Date_transaction, TICKET_ID)

Weather (idWeatherDate, max_temperature, min_temperature, avg_temperature, max_humidex, rainMM, snowCM, snow_on_groundCM, sunrise_time, sunset_time, total_daylightHR)

6.0 Queries

6.1 Query 1

Objective: To see if ride usage is reduced when there is precipitation. If there is a forecast for precipitation, it can help the park plan for staffing accordingly.

Assumptions: Want to include if a customer rode on a ride more than once on the same day to see if that matters when there is precipitation. Precipitation is defined as having rained or snowed (or both) on a given day).

Code:

```
select "Ride usage during no precipitation" as "Precipitation vs No Precipitation",
NumUsageNoPrecipitation "Ride usage",
concat(round(NumUsageNoPrecipitation/(NumUsageNoPrecipitation+NumUsagePrecipitation)*100,2),"
%") as "Percentage of total ride usage"
from
```

```
        (SELECT count(*) as NumUsageNoPrecipitation
        FROM rides_tickets, weather
        where weather.idWeatherDate=rides_tickets.idWeatherDate and ((rainMM/10)+snowCM)=0) as
        UsageNoPrecipitation,

        (SELECT count(*) as NumUsagePrecipitation
        FROM rides_tickets, weather
        where weather.idWeatherDate=rides_tickets.idWeatherDate and ((rainMM/10)+snowCM)>0) as
        UsagePrecipitation

union

select "Ride usage during precipitation" as "Precipitation vs No Precipitation", NumUsagePrecipitation
"Ride usage",
concat(round(NumUsagePrecipitation/(NumUsageNoPrecipitation+NumUsagePrecipitation)*100,2),"%")
as "Percentage of total ride usage"
from
        (SELECT count(*) as NumUsageNoPrecipitation
        FROM rides_tickets, weather
        where weather.idWeatherDate=rides_tickets.idWeatherDate and ((rainMM/10)+snowCM)=0) as
        UsageNoPrecipitation,

        (SELECT count(*) as NumUsagePrecipitation
        FROM rides_tickets, weather
        where weather.idWeatherDate=rides_tickets.idWeatherDate and ((rainMM/10)+snowCM)>0) as
        UsagePrecipitation;
```

Output:

	Precipitation vs No Precipitation	Ride usage	Percentage of total ride usage
▶	Ride usage during no precipitation	2458	61.47%
	Ride usage during precipitation	1541	38.53%

Interpretation: Precipitation does have a significant impact on the usage of rides and La Ronde should plan their staff accordingly.

6.2 Query 2

Objective: To see if precipitation has an impact on total sales revenue from facilities.

Assumptions: Precipitation is defined as having rained or snowed (or both) on a given day.

Code:

```
select "Facility sales during no precipitation" as "Precipitation vs No Precipitation",
concat("$",round(SalesNoPrecipitation,2)) "Facility Sales",
concat(round(SalesNoPrecipitation/(SalesNoPrecipitation+SalesPrecipitation)*100,2),"%") as
"Percentage of total Sales"
from
        (SELECT sum(quantity*price) as SalesNoPrecipitation
```

```
        FROM facility_tickets, weather, items, purchases
        where weather.idWeatherDate=facility_tickets.idWeatherDate and
        date(purchases.date_transaction)=facility_tickets.idWeatherDate and
        items.ItemID=purchases.ItemID and ((rainMM/10)+snowCM)=0) as SalesNoPrecipitation,

        (SELECT sum(quantity*price) as SalesPrecipitation
        FROM facility_tickets, weather, items, purchases
        where weather.idWeatherDate=facility_tickets.idWeatherDate and
        date(purchases.date_transaction)=facility_tickets.idWeatherDate and
        items.ItemID=purchases.ItemID and ((rainMM/10)+snowCM)>0) as SalesPrecipitation

union

select "Facility sales during precipitation" as "Precipitation vs No Precipitation", concat("$",
round(SalesPrecipitation,2)) "Facility Sales",
concat(round(SalesPrecipitation/(SalesNoPrecipitation+SalesPrecipitation)*100,2),"%") as "Percentage of
total Sales"
from
        (SELECT sum(quantity*price) as SalesNoPrecipitation
        FROM facility_tickets, weather, items, purchases
        where weather.idWeatherDate=facility_tickets.idWeatherDate and
        date(purchases.date_transaction)=facility_tickets.idWeatherDate and
        items.ItemID=purchases.ItemID and ((rainMM/10)+snowCM)=0) as SalesNoPrecipitation,

        (SELECT sum(quantity*price) as SalesPrecipitation
        FROM facility_tickets, weather, items, purchases
        where weather.idWeatherDate=facility_tickets.idWeatherDate and
        date(purchases.date_transaction)=facility_tickets.idWeatherDate and
        items.ItemID=purchases.ItemID and ((rainMM/10)+snowCM)>0) as SalesPrecipitation;
```

Output:

	Precipitation vs No Precipitation	Facility Sales	Percentage of total Sales
▶	Facility sales during no precipitation	$998780.17	61.66%
	Facility sales during precipitation	$620972.62	38.34%

Interpretation: Sales at the park are significantly impacted by precipitation. This goes in line with the previous query (6.1). For the park to save on staffing costs, they don't have to have full staff on days where the forecast includes precipitation.

6.3 Query 3

Objective: To see how much revenue was made by year, month, and various sales.

Code:

```
select year(Date_transaction) as Year, case
        when month(Date_transaction)=1 then "January"
        when month(Date_transaction)=2 then "February"
        when month(Date_transaction)=3 then "March"
```

```
        when month(Date_transaction)=4 then "April"
        when month(Date_transaction)=5 then "May"
        when month(Date_transaction)=6 then "June"
        when month(Date_transaction)=7 then "July"
        when month(Date_transaction)=8 then "August"
        when month(Date_transaction)=9 then "September"
        when month(Date_transaction)=10 then "October"
        when month(Date_transaction)=11 then "November"
        else "December"
end as Month ,concat("$",round(sum(ticket.price),2)) as "Total Ticket Sales",
concat(round(sum(ticket.price)/(sum(ticket.price)+sum(items.price*quantity))*100,2),"%") as
"Percentage of total revenue from ticket sales", concat("$",round(sum(items.price*quantity),2)) as "Total
Facility Sales",
concat(round(sum(items.price*quantity)/(sum(ticket.price)+sum(items.price*quantity))*100,2),"%") as
"Percentage of total revenue from facility sales",
concat("$",round(sum(ticket.price)+sum(items.price*quantity),2)) as "Total Revenue"
from ticket, items, purchases
where items.ItemID=purchases.ItemID
group by Year, Month
order by "Total Revenue";
```

Output:

Year	Month	Total Ticket Sales	Percentage of total revenue from ticket sales	Total Facility Sales	Percentage of total revenue from facility sales	Total Revenue
2019	December	$144757080	93.23%	$10509089.75	6.77%	$155266169.75
2020	April	$135168470	93.23%	$9818089.77	6.77%	$144986559.77
2020	June	$141663980	93.16%	$10407629.76	6.84%	$152071609.76
2020	March	$140426740	93.68%	$9480689.77	6.32%	$149907429.77
2020	January	$130219510	92.67%	$10299489.78	7.33%	$140518999.78
2020	February	$122177450	93.69%	$8224599.81	6.31%	$130402049.81
2019	November	$103618850	93.36%	$7375489.82	6.64%	$110994339.82
2020	July	$118465730	92.88%	$9082459.8	7.12%	$127548189.8
2020	May	$124342620	93.09%	$9231159.77	6.91%	$133573779.77
2020	August	$49489600	93.23%	$3591819.92	6.77%	$53081419.92

Interpretation: December 2019 made La Ronde the most money. The decline in revenue can be linked to the pandemic with a decrease number in park attendance.

6.4 Query 4

Objective: Building on the query that looks at facilities near rides (section 3.3), the purpose of this query is to see how much revenue was made from those facilities and to see if being near a ride results with more revenue.

Assumptions: Using the view table “facilityrideproximity” from section 3.3.

Code:

```
SELECT Rideproximity, Facility_type, concat("$",round(sum(quantity*price),2)) TotalRevenue
FROM facilityrideproximity, items, purchases
where facilityrideproximity.facility_id=items.facility_id and items.ItemID=purchases.ItemID
group by Rideproximity,Facility_type
order by sum(quantity*price) desc;
```

Output:

	RideProximity	FACILITY_TYPE	TotalRevenue
▶	Near Ride	Dinning	$37183.79
	Not near a ride	Dinning	$21752.32
	Near Ride	Shopping	$14961.27
	Not near a ride	Shopping	$14123.14

Interpretation: Facilities, whether for shopping or dinning, make more revenue than facilities that are not near rides. Having dining facilities near rides makes La Ronde the most revenue.

6.5 Query 5

Objective: See the top 10 customers that spend the most and reward them with discounts for customer loyalty.

Code:

```
select concat(first_name, " ", last_name) as Name,
concat("$",round(sum(purchases.quantity*items.price),2)) as FacilitySpend
from customer, ticket, items, purchases
where items.ItemID=purchases.ItemID and customer.C_ID=ticket.C_ID and
ticket.TICKET_ID=purchases.TICKET_ID
group by purchases.TICKET_ID
order by sum(purchases.quantity*items.price) desc limit 10;
```

Output:

	Name	FacilitySpend
▶	Jermain Meachem	$321.75
	Walden Franzoli	$296.17
	Mirna Questier	$272.8
	Gregorio Wetherill	$271.3
	Kati Delhanty	$268.85
	Filippo Tosdevin	$264.77
	Nicolais Boycott	$262.77
	Addie Garratty	$262.7
	Nara Girk	$260.87
	Gregorio Wetherill	$257.82

Interpretation: Jermain Meachem is the top spending customer at our facilities. La Ronde may want to reward these customers by giving them discounts at their favourite facilities for when they return to the park.

6.6 Query 6 (Bonus Query)

Objective: See if temperature influences revenue made at facilities.

Assumptions: It's considered "cold" when the average temperature is below 0ºC, "chilly" when the average temperature is below 10ºC, "comfortable" when the average temperature is below 20ºC, else "hot" (based off of what I feel is cold, chilly, comfortable, and hot).

Code:

```
/* Step 1 - Create a view table with the "temperature feeling" based on the average temperature*/
create view tempfeeling as
select idWeatherDate, avg_temperature, total_daylightHR,
case
        when avg_temperature<=0 then "Cold"
        when avg_temperature<=10 then "Chilly"
        when avg_temperature<=20 then "Comfortable"
        else "Hot"
end as temp_feeling
from weather;

/* Step 2 - See which "temperature feeling" makes the most revenue from facilities*/
select temp_feeling, concat("$",round(sum(purchases.quantity*items.price),2)) as "Total Revenue"
from tempfeeling, facility_tickets, ticket, purchases, items
where tempfeeling.idWeatherDate=facility_tickets.idWeatherDate and
facility_tickets.TICKET_ID=ticket.TICKET_ID and ticket.TICKET_ID=purchases.TICKET_ID and
items.ItemID=purchases.ItemID
group by temp_feeling
order by sum(purchases.quantity*items.price) desc;
```

Output:

From the view table “tempfeeling”:

idWeatherDate	avg_temperature	total_daylightHR	temp_feeling
2019-01-01	-4.95	8.77	Cold
2019-01-02	-12.5	8.78	Cold
2019-01-03	-7.8	8.8	Cold
2019-01-04	0.4	8.82	Chilly
2019-01-05	1.14	8.83	Chilly
2019-01-06	-5.85	8.85	Cold

From the final output:

temp_feeling	Total Revenue
Cold	$163630.57
Chilly	$119586.46
Hot	$104544.86
Comfortable	$48248.99

Interpretation: Surprisingly, “cold” temperatures lead to the most revenue generated from facilities. La Ronde may want to create seasonal events to help boost revenues during other weather types.

Appendix

Refer to the .sql file for create tables, insert statements, and queries.