Individual Project Part 1

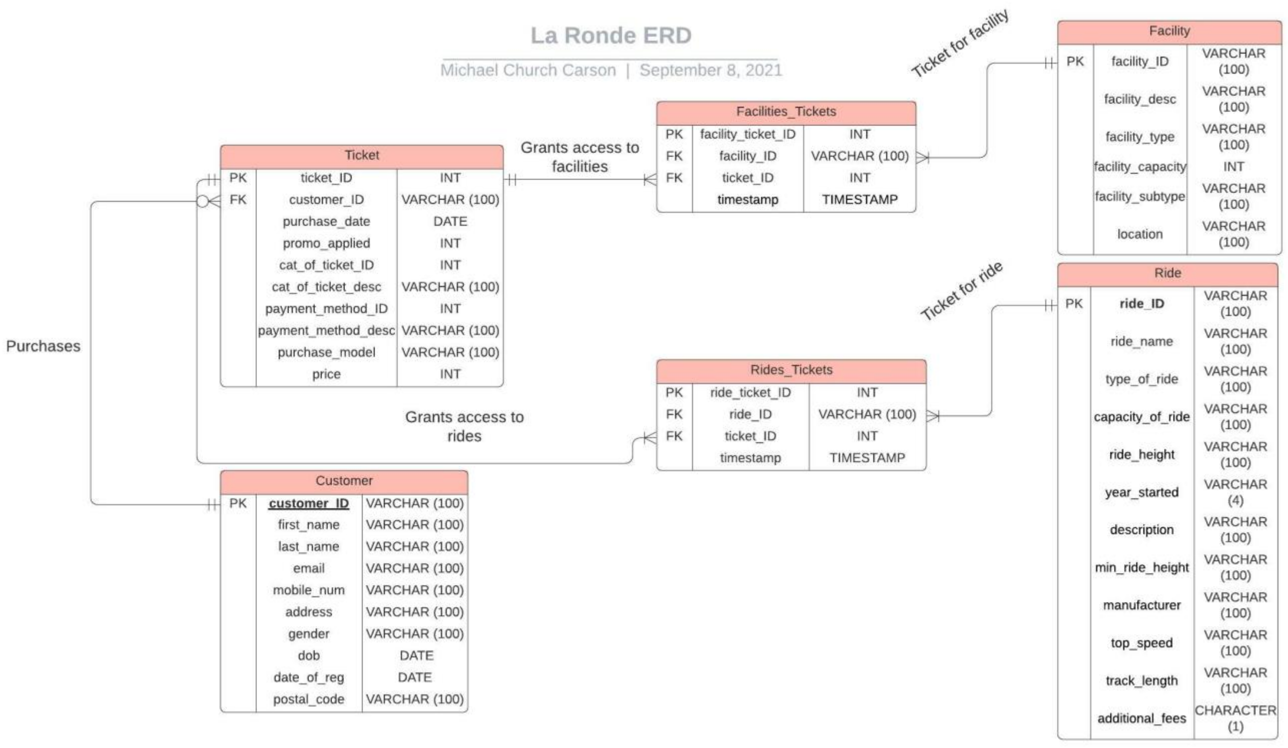
Report #1: Tasks 1 to 4
INSY 661

NOTE: All required SQL statements for Tasks 1 to 4 are saved in the plain text file: "Individual Project Part1_Report1_Task1-4.sql"

There is also a .sql file called "Exported Tables from phpMyAdmin" that was included in my submission, it contains the table structure for tasks 1 to 5.

Tasks 1 to 4:

1) ERD



2) RELATIONAL MODEL

Customer (<u>customer_ID</u>, first_name, last_name, email, mobile_num, address, gender, dob, date_of_reg, postal_code)

Facility (<u>facility_ID</u>, facility_desc, facility_type, facility_capacity, facility_subtype, location)

Ride (<u>ride_ID</u>, ride_name, type_of_ride, capacity_of_ride, ride_height, year_started, description, min_ride_height, manufacturer, top_speed, track_length, additional_fees)

Ticket (<u>ticket_ID</u>, <u>customer_ID</u>, purchase_date, promo_applied, cat_of_ticket_ID, cat_of_ticket_desc, payment_method_ID, payment_method_desc, purchase_model, price)

Rides_Tickets (<u>ride_ticket_ID</u>, <u>ride_ID</u>, <u>ticket_ID</u>, timestamp)

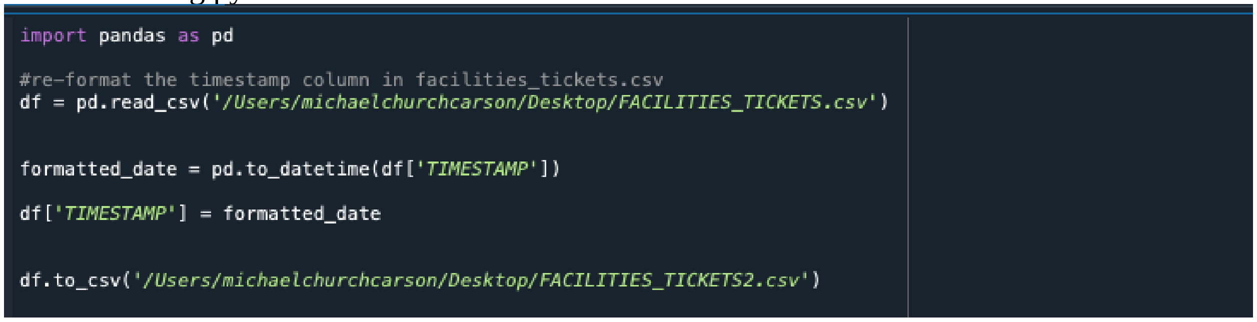
Facilities_Tickets (<u>facility_ticket_ID</u>, <u>facility_ID</u>, <u>ticket_ID</u>, timestamp)

DDL: Please refer to "Individual Project Part1_Report1_Task1-4.sql" for the SQL used to create the tables

3) POPULATE TABLES

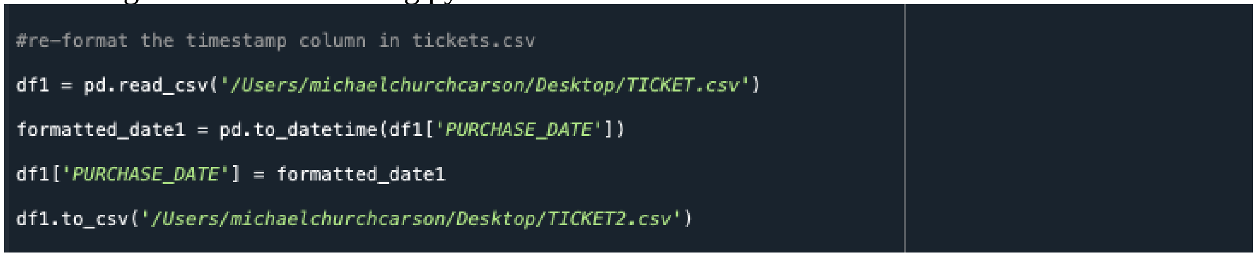
Before populating the tables using the csv files, six changes that must be made to clean the date in the csv files provided:

1. A new csv file called FACILITIES_TICKETS must be created. This new csv contains the FACILITY_ID, TICKET_ID, and TIMESTAMP columns from the original FACILITY.csv file.
2. "null" needs to be inserted into all the blank cells in the RIDE.csv file. Since there are only 43 rows in the RIDE.csv file, this was just done manually in excel.
3. In the FACILITIES_TICKETS csv, the format of the entries in the TIMESTAMP column is incorrect for inserting into the relational tables as a MySQL timestamp variable type. They are formatted as "8/01/2020 12:31", but they need to be formatted as YYYY-MM-DD HH:MM. The formatting can be fixed using the pandas module in python. The below screen shot shows how this formatting issue was fixed using python.



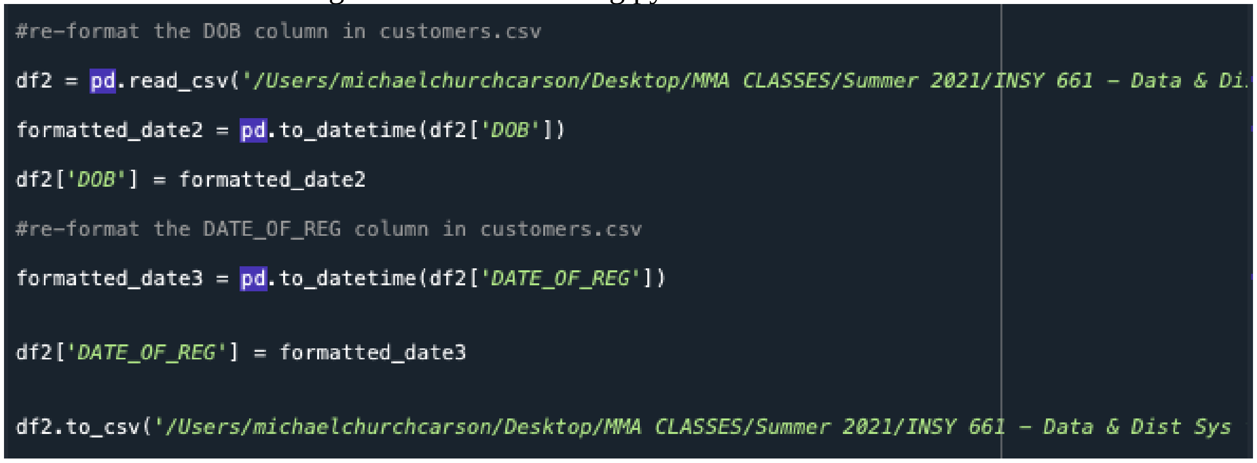
```
import pandas as pd

#re-format the timestamp column in facilities_tickets.csv
df = pd.read_csv('/Users/michaelchurchcarson/Desktop/FACILITIES_TICKETS.csv')


formatted_date = pd.to_datetime(df['TIMESTAMP'])

df['TIMESTAMP'] = formatted_date


df.to_csv('/Users/michaelchurchcarson/Desktop/FACILITIES_TICKETS2.csv')
```

4. The timestamp format of the entries in the PURCHASE_DATE column in TICKETS.csv are incorrect. They are formatted as "DD-MM-YYYY", but they need to be formatted as YYYY-MM-DD HH:MM. The below screen shot shows how this formatting issue was fixed using python.



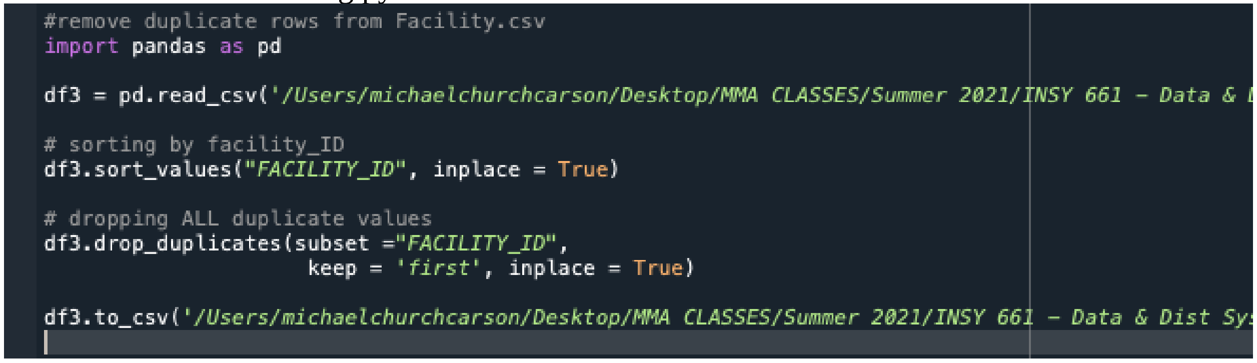
```
#re-format the timestamp column in tickets.csv

df1 = pd.read_csv('/Users/michaelchurchcarson/Desktop/TICKET.csv')

formatted_date1 = pd.to_datetime(df1['PURCHASE_DATE'])

df1['PURCHASE_DATE'] = formatted_date1

df1.to_csv('/Users/michaelchurchcarson/Desktop/TICKET2.csv')
```

5. The DOB and DATE_OF_REG columns in the CUSTOMERS.csv file need to be re-formatted into the YYYY-MM-DD HH:MM formatting. The below screen shot shows how this formatting issue was fixed using python.



```
#re-format the DOB column in customers.csv

df2 = pd.read_csv('/Users/michaelchurchcarson/Desktop/MMA CLASSES/Summer 2021/INSY 661 - Data & Di

formatted_date2 = pd.to_datetime(df2['DOB'])

df2['DOB'] = formatted_date2

#re-format the DATE_OF_REG column in customers.csv

formatted_date3 = pd.to_datetime(df2['DATE_OF_REG'])


df2['DATE_OF_REG'] = formatted_date3


df2.to_csv('/Users/michaelchurchcarson/Desktop/MMA CLASSES/Summer 2021/INSY 661 - Data & Dist Sys
```

6. The FACILITY.csv file contains duplicate rows for every facility. All the duplicate rows need to be removed such that the FACILITY.csv file only contains one unique row for each facility at La Ronde. The below screen shot shows how the duplicate rows can be removed using python.



```
#remove duplicate rows from Facility.csv
import pandas as pd

df3 = pd.read_csv('/Users/michaelchurchcarson/Desktop/MMA CLASSES/Summer 2021/INSY 661 - Data & I

# sorting by facility_ID
df3.sort_values("FACILITY_ID", inplace = True)

# dropping ALL duplicate values
df3.drop_duplicates(subset ="FACILITY_ID",
                     keep = 'first', inplace = True)

df3.to_csv('/Users/michaelchurchcarson/Desktop/MMA CLASSES/Summer 2021/INSY 661 - Data & Dist Sys
```

After these 6 data cleaning steps, the csv files can now be imported to populate the tables created in the task 2. After the csv files we imported as new tables, the first row of each table contained the column names as if they were data. To fix this, the first row was deleted from each table. The below screen shot shows the 12 tables that are now in the "LaRonde" database. Six of the tables were generated from the relational schema using DLL during task 2, and six were generated by importing the csv files. The tables generated by importing the csv files have the suffix "CSV".

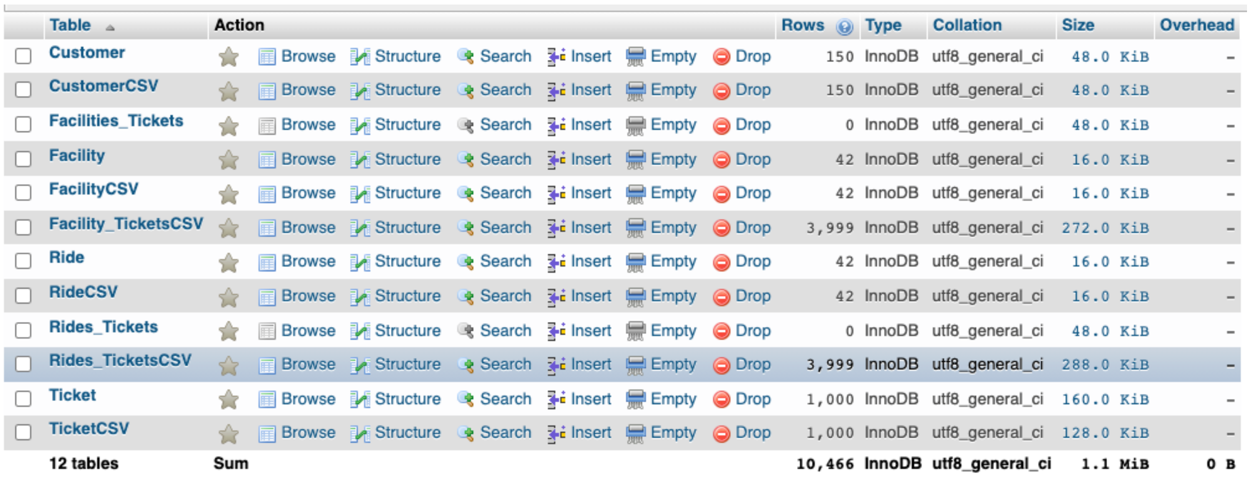
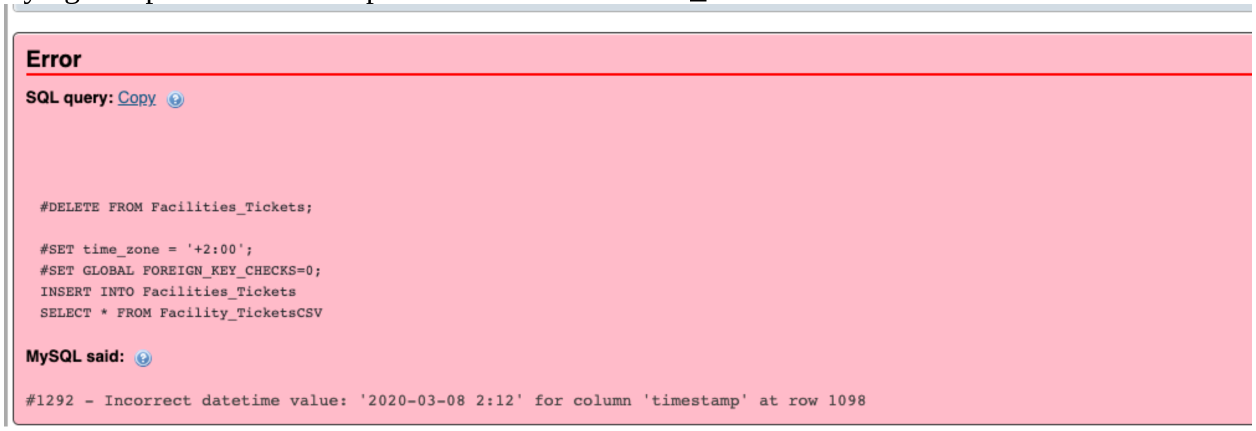


Table	Action						Rows	Type	Collation	Size	Overhead
Customer	Browse	Structure	Search	Insert	Empty	Drop	150	InnoDB	utf8_general_ci	48.0 KiB	-
CustomerCSV	Browse	Structure	Search	Insert	Empty	Drop	150	InnoDB	utf8_general_ci	48.0 KiB	-
Facilities_Tickets	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_general_ci	48.0 KiB	-
Facility	Browse	Structure	Search	Insert	Empty	Drop	42	InnoDB	utf8_general_ci	16.0 KiB	-
FacilityCSV	Browse	Structure	Search	Insert	Empty	Drop	42	InnoDB	utf8_general_ci	16.0 KiB	-
Facility_TicketsCSV	Browse	Structure	Search	Insert	Empty	Drop	3,999	InnoDB	utf8_general_ci	272.0 KiB	-
Ride	Browse	Structure	Search	Insert	Empty	Drop	42	InnoDB	utf8_general_ci	16.0 KiB	-
RideCSV	Browse	Structure	Search	Insert	Empty	Drop	42	InnoDB	utf8_general_ci	16.0 KiB	-
Rides_Tickets	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_general_ci	48.0 KiB	-
Rides_TicketsCSV	Browse	Structure	Search	Insert	Empty	Drop	3,999	InnoDB	utf8_general_ci	288.0 KiB	-
Ticket	Browse	Structure	Search	Insert	Empty	Drop	1,000	InnoDB	utf8_general_ci	160.0 KiB	-
TicketCSV	Browse	Structure	Search	Insert	Empty	Drop	1,000	InnoDB	utf8_general_ci	128.0 KiB	-
12 tables	Sum						10,466	InnoDB	utf8_general_ci	1.1 MiB	0 B

The six tables from the csv files can now be used to populate the 6 tables from the relational schema.

Please refer to "Individual Project Part1_Report1_Task1-4.sql" to see the SQL used to populate the relational schema tables.

Note: You will see in the .sql file that before inputting the data from the Facility_TicketsCSV to the Facilities_Tickets table, row 1098 from the Facility_TicketsCSV table needs to be deleted. This is because the TIMESTAMP value of row 1098 in the Facility_TicketsCSV is "2020-03-08 2:12:00", and daylight savings time began in the EST time zone on March 8, 2020, at 2 AM. Therefore, the clocks were moved forward by an hour at 2 AM, and consequently, the timestamp of "2020-03-08 2:12:00" does not exist. MySQL returns an error (see below screenshot) when trying to input this timestamp value into the Facilities_Tickets table.



Error

SQL query: Copy

```
#DELETE FROM Facilities_Tickets;

#SET time_zone = '+2:00';
#SET GLOBAL FOREIGN_KEY_CHECKS=0;
INSERT INTO Facilities_Tickets
SELECT * FROM Facility_TicketsCSV
```

MySQL said:

```
#1292 - Incorrect datetime value: '2020-03-08 2:12' for column 'timestamp' at row 1098
```

This error is resolved by removing this row from the Facility_TicketsCSV table. This means there is a loss of one row of data, but losing this one row of data that has a non-existent timestamp is better than adjusting the time zone for the entire table, which would change the timestamp values in every row.

4) PREPARE 10 SQL QUERIES:

Before preparing SQL queries, the 6 tables that were created when the csv files were imported can be removed. Now only the 6 tables from the relational schema are remaining in the LaRonde database. See below screenshot.

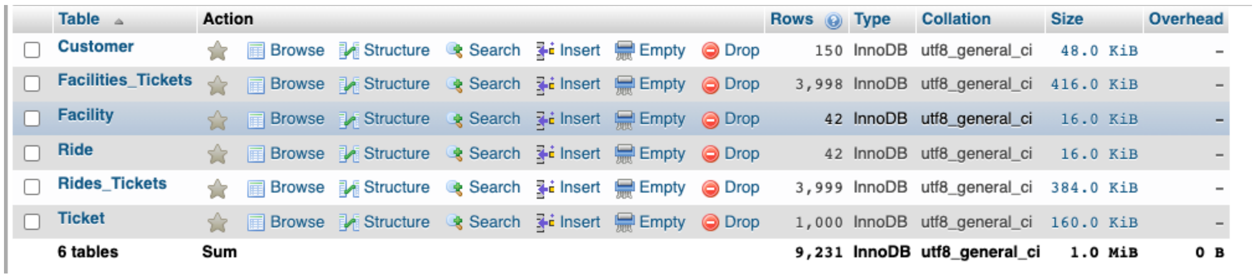


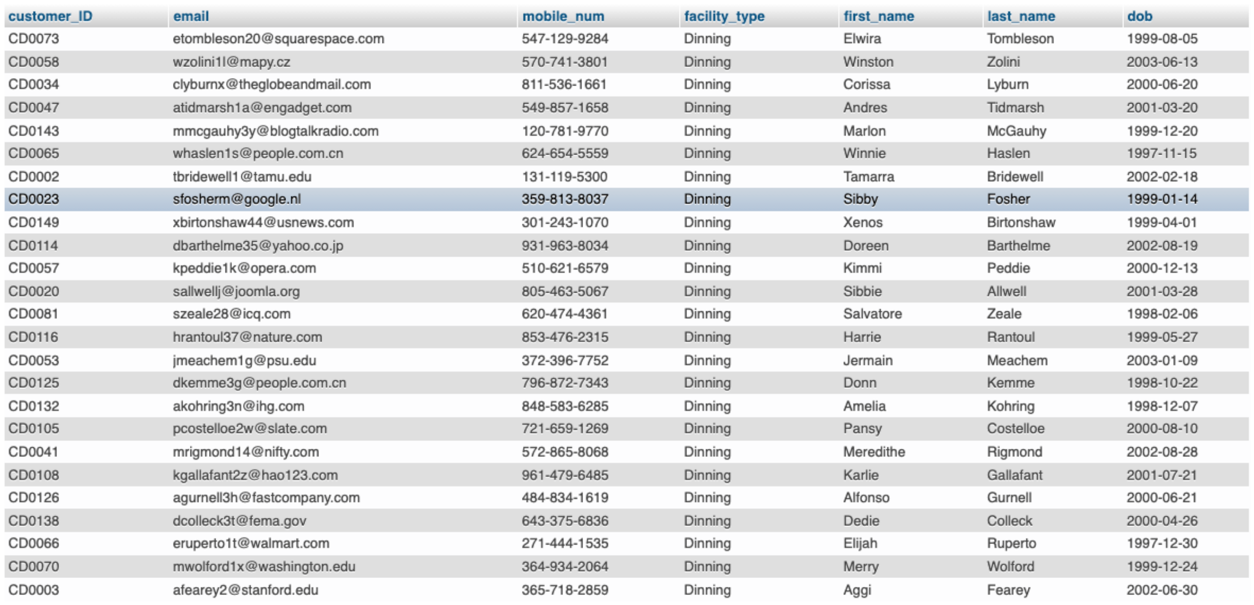
Table	Action							Rows	Type	Collation	Size	Overhead
Customer	☆	Browse	Structure	Search	Insert	Empty	Drop	150	InnoDB	utf8_general_ci	48.0 KiB	–
Facilities_Tickets	☆	Browse	Structure	Search	Insert	Empty	Drop	3,998	InnoDB	utf8_general_ci	416.0 KiB	–
Facility	☆	Browse	Structure	Search	Insert	Empty	Drop	42	InnoDB	utf8_general_ci	16.0 KiB	–
Ride	☆	Browse	Structure	Search	Insert	Empty	Drop	42	InnoDB	utf8_general_ci	16.0 KiB	–
Rides_Tickets	☆	Browse	Structure	Search	Insert	Empty	Drop	3,999	InnoDB	utf8_general_ci	384.0 KiB	–
Ticket	☆	Browse	Structure	Search	Insert	Empty	Drop	1,000	InnoDB	utf8_general_ci	160.0 KiB	–
6 tables	Sum							9,231	InnoDB	utf8_general_ci	1.0 MiB	0 B

Query 1:
Objective: La Ronde wants to start a student discount program for its dinning facilities to encourage students to eat at the amusement park. To promote this new student dinning discount, La Ronde wants the contact information (email address and phone number) of all the park customers who have used a dinning facility *and* qualify as students.
Assumption: To qualify as a student, customers must be between 18 and 25 years old (as of today, September 8, 2021).

Code:

```
SELECT DISTINCT Customer.customer_ID, email, mobile_num, facility_type,
Customer.first_name, Customer.last_name, dob
FROM Facility, Facilities_Tickets, Ticket, Customer
WHERE Facility.facility_ID = Facilities_Tickets.facility_ID AND
Facilities_Tickets.ticket_ID = Ticket.ticket_ID AND
Ticket.customer_ID = Customer.customer_ID AND
facility_type = "Dinning" AND
dob BETWEEN '1996-09-08' AND '2003-09-08';
```

Output Screenshot:



customer_ID	email	mobile_num	facility_type	first_name	last_name	dob
CD0073	etombleson20@squarespace.com	547-129-9284	Dinning	Elwira	Tombleson	1999-08-05
CD0058	wzolini1l@mapy.cz	570-741-3801	Dinning	Winston	Zolini	2003-06-13
CD0034	clyburnx@theglobeandmail.com	811-536-1661	Dinning	Corissa	Lyburn	2000-06-20
CD0047	atidmarsh1a@engadget.com	549-857-1658	Dinning	Andres	Tidmarsh	2001-03-20
CD0143	mmcgauhy3y@blogtalkradio.com	120-781-9770	Dinning	Marlon	McGauhy	1999-12-20
CD0065	whaslen1s@people.com.cn	624-654-5559	Dinning	Winnie	Haslen	1997-11-15
CD0002	tbridewell1@tamu.edu	131-119-5300	Dinning	Tamarra	Bridewell	2002-02-18
CD0023	sfosherm@google.nl	359-813-8037	Dinning	Sibby	Fosher	1999-01-14
CD0149	xbirtonshaw44@usnews.com	301-243-1070	Dinning	Xenos	Birtonshaw	1999-04-01
CD0114	dbarthelme35@yahoo.co.jp	931-963-8034	Dinning	Doreen	Barthelme	2002-08-19
CD0057	kpeddie1k@opera.com	510-621-6579	Dinning	Kimmi	Peddie	2000-12-13
CD0020	sallwellj@joomla.org	805-463-5067	Dinning	Sibbie	Allwell	2001-03-28
CD0081	szeale28@icq.com	620-474-4361	Dinning	Salvatore	Zeale	1998-02-06
CD0116	hrantoul37@nature.com	853-476-2315	Dinning	Harrie	Rantoul	1999-05-27
CD0053	jmeachem1g@psu.edu	372-396-7752	Dinning	Jermain	Meachem	2003-01-09
CD0125	dkemme3g@people.com.cn	796-872-7343	Dinning	Donn	Kemme	1998-10-22
CD0132	akohring3n@ihg.com	848-583-6285	Dinning	Amelia	Kohring	1998-12-07
CD0105	pcostelloe2w@slate.com	721-659-1269	Dinning	Pansy	Costelloe	2000-08-10
CD0041	mrigmond14@nifty.com	572-865-8068	Dinning	Meredithe	Rigmond	2002-08-28
CD0108	kgallafant2z@hao123.com	961-479-6485	Dinning	Karlie	Gallafant	2001-07-21
CD0126	agurnell3h@fastcompany.com	484-834-1619	Dinning	Alfonso	Gurnell	2000-06-21
CD0138	dcolleck3t@fema.gov	643-375-6836	Dinning	Dedie	Colleck	2000-04-26
CD0066	eruperto1t@walmart.com	271-444-1535	Dinning	Elijah	Ruperto	1997-12-30
CD0070	mwolford1x@washington.edu	364-934-2064	Dinning	Merry	Wolford	1999-12-24
CD0003	afearey2@stanford.edu	365-718-2859	Dinning	Aggi	Fearey	2002-06-30

As this screenshot of the partial output shows, this query displays the name and contact information of all the customers who have attended a dinning facility and who qualify as students.

Query 2:
Objective: La Ronde is introducing a new student discount program for its annual passes (non-promo) in 2022. The park's management would like to give students a 20% discount on annual passes in 2022, but before it does so it would like to better understand how this change will decrease its total revenue (in terms of dollars).

To help management, examine the last full year of ticket sales (i.e. 2020), and show how much revenue would have been lost if there had been a 20% discount for students who bought full-price annual passes.

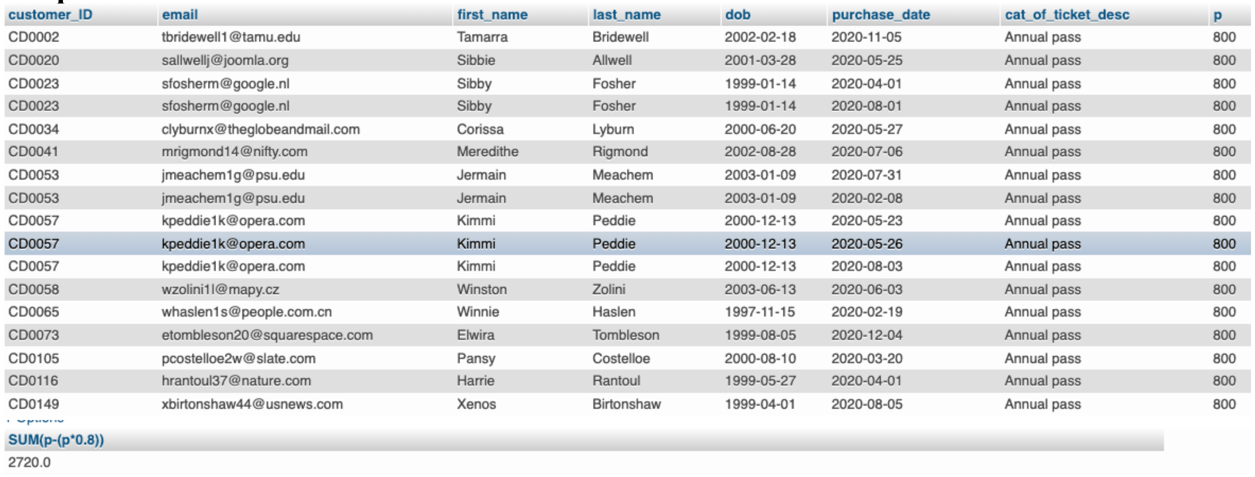
Assumption: To qualify as a student, customers must be between 18 and 25 years old (as of today, September 8, 2021). The student discount would only apply to annual passes purchased at full price (i.e., with no promo-code).

Code:

```
CREATE VIEW studentdiscount AS
SELECT DISTINCT Customer.customer_ID, email, Customer.first_name, Customer.last_name,
dob, Ticket.purchase_date, Ticket.cat_of_ticket_desc, Ticket.price as p
FROM Ticket, Customer
WHERE Ticket.customer_ID = Customer.customer_ID AND
Ticket.promo_applied = 0 AND
Ticket.cat_of_ticket_desc = "Annual Pass" AND
Ticket.purchase_date > '2019-12-31' AND
dob BETWEEN '1996-09-08' AND '2003-09-08';

SELECT SUM(p)
```

```
FROM studentdiscount;

SELECT SUM(p-(p*0.8))
FROM studentdiscount
```

Output Screenshot:



customer_ID	email	first_name	last_name	dob	purchase_date	cat_of_ticket_desc	p
CD0002	tbridewell1@tamu.edu	Tamarra	Bridewell	2002-02-18	2020-11-05	Annual pass	800
CD0020	sallwellj@joomla.org	Sibbie	Allwell	2001-03-28	2020-05-25	Annual pass	800
CD0023	sfosherm@google.nl	Sibby	Fosher	1999-01-14	2020-04-01	Annual pass	800
CD0023	sfosherm@google.nl	Sibby	Fosher	1999-01-14	2020-08-01	Annual pass	800
CD0034	clyburnx@theglobeandmail.com	Corissa	Lyburn	2000-06-20	2020-05-27	Annual pass	800
CD0041	mrigmond14@nifty.com	Meredithe	Rigmond	2002-08-28	2020-07-06	Annual pass	800
CD0053	jmeachem1g@psu.edu	Jermain	Meachem	2003-01-09	2020-07-31	Annual pass	800
CD0053	jmeachem1g@psu.edu	Jermain	Meachem	2003-01-09	2020-02-08	Annual pass	800
CD0057	kpeddie1k@opera.com	Kimmi	Peddie	2000-12-13	2020-05-23	Annual pass	800
CD0057	kpeddie1k@opera.com	Kimmi	Peddie	2000-12-13	2020-05-26	Annual pass	800
CD0057	kpeddie1k@opera.com	Kimmi	Peddie	2000-12-13	2020-08-03	Annual pass	800
CD0058	wzolini1l@mapy.cz	Winston	Zolini	2003-06-13	2020-06-03	Annual pass	800
CD0065	whaslen1s@people.com.cn	Winnie	Haslen	1997-11-15	2020-02-19	Annual pass	800
CD0073	etombleson20@squarespace.com	Elwira	Tombleson	1999-08-05	2020-12-04	Annual pass	800
CD0105	pcostelloe2w@slate.com	Pansy	Costelloe	2000-08-10	2020-03-20	Annual pass	800
CD0116	hrantoul37@nature.com	Harrie	Rantoul	1999-05-27	2020-04-01	Annual pass	800
CD0149	xbirtonshaw44@usnews.com	Xenos	Birtonshaw	1999-04-01	2020-08-05	Annual pass	800

```
SUM(p-(p*0.8))
2720.0
```

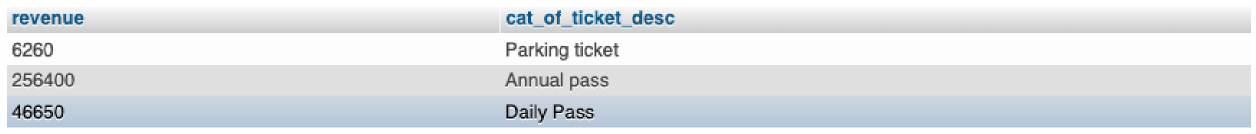
From the above outputs we can see that in 2020, if students who bought full price annual passes (non-promo) were offered a 20% discount, La Ronde would have lost $2720 in revenue.

Query 3:
Objective: What percentage of La Ronde's total ticket revenue was from selling parking tickets?
Assumption: Parking ticket refers to a ticket customers buy if they want to park their car at La Ronde while visiting, and not a ticket that is given for parking illegally.

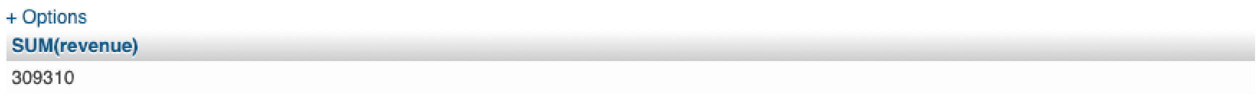
Code:

```
CREATE VIEW total_revenue AS
SELECT SUM(price) AS revenue, cat_of_ticket_desc
FROM Ticket
WHERE cat_of_ticket_desc = "Parking ticket"
UNION
SELECT SUM(price), cat_of_ticket_desc
FROM Ticket
WHERE cat_of_ticket_desc = "Annual pass"
UNION
SELECT SUM(price), cat_of_ticket_desc
FROM Ticket
WHERE cat_of_ticket_desc = "Daily pass";

SELECT SUM(revenue)
FROM total_revenue;
```

Output Screenshot:



revenue	cat_of_ticket_desc
6260	Parking ticket
256400	Annual pass
46650	Daily Pass



+ Options

SUM(revenue)
309310

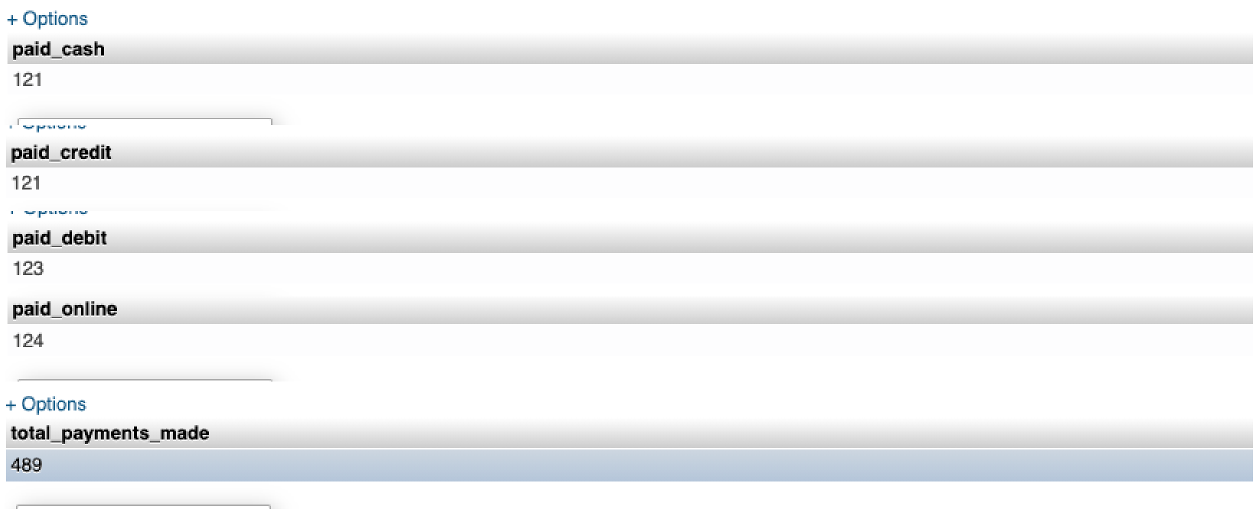
From the above outputs we can see that the percentage of total ticket revenue from parking tickets is equal to (6260/309310)*100 = 2.02%.

Query 4:
Objective: What is the most popular payment method (cash, credit, debit, or online), and what is the least popular payment method? Answer this query by showing the total number of customers that used each payment method.
Assumption: Popularity here is determined by only counting each customer's choice of payment method *once*. Even if one customer pays using cash 20 times, they would only add one to the count of customers who pay in cash.

Code:

```
CREATE VIEW payment_methods AS
SELECT DISTINCT Ticket.customer_ID, payment_method_desc
FROM Ticket;

SELECT COUNT(payment_methods.customer_ID) AS paid_cash
FROM payment_methods
WHERE payment_method_desc = "Cash";

SELECT COUNT(payment_methods.customer_ID) AS paid_credit
FROM payment_methods
WHERE payment_method_desc = "Credit Card";

SELECT COUNT(payment_methods.customer_ID) AS paid_debit
FROM payment_methods
WHERE payment_method_desc = "Debit Card";

SELECT COUNT(payment_methods.customer_ID) AS paid_online
FROM payment_methods
WHERE payment_method_desc = "Online Payment";

SELECT COUNT(payment_methods.customer_ID) AS total_payments_made
FROM payment_methods;
```

Output Screenshot:



```
+ Options
paid_cash
121
paid_credit
121
paid_debit
123
paid_online
124
+ Options
total_payments_made
489
```

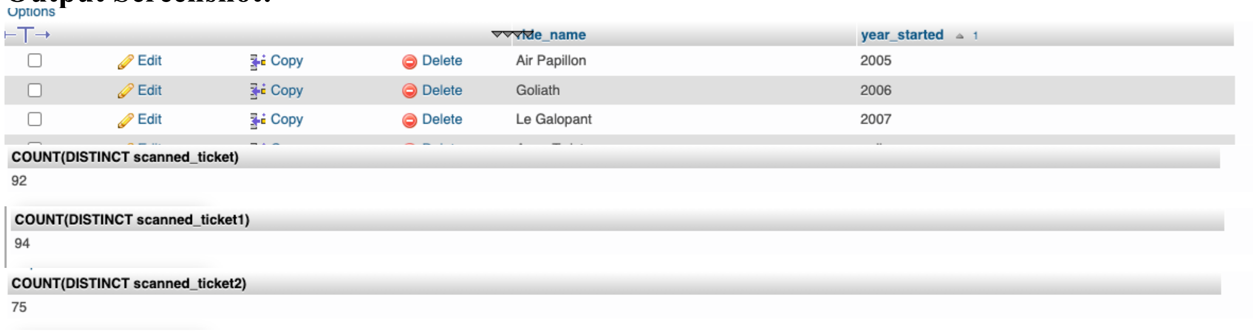
From the above outputs we can see that 24.7% (121/489) of payments we made in cash, 24.7% (121/489) were made in credit, 25.2% (123/489) in debit, and 25.4% (124/489) were made online. Therefore, online payment was the most popular, and cash and credit tied for the least popular.

Query 5:
Objective: Find all the rides that have started operating since the year 2000 and show which one of these rides is the most popular.
Assumptions: The year_started column in the Ride table indicates the year that a ride started operating, and if there is no year included in this column for a particular ride, then the year that this ride started operating is unknown. Popularity is determined by the total number of *unique* customers who take the ride. A ride is considered as taken when a customer has scanned his/her ticket and as a result, a ticket is associated with a ride.

Code:
```
SELECT ride_name, year_started
FROM Ride
WHERE year_started > '2000'
ORDER BY year_started;

CREATE VIEW newride05 AS
SELECT DISTINCT Rides_Tickets.ticket_ID AS scanned_ticket
FROM Ride, Rides_Tickets
WHERE Ride.ride_ID = Rides_Tickets.ride_ID AND
year_started = '2005';

SELECT COUNT(DISTINCT scanned_ticket)
FROM newride05;

CREATE VIEW newride06 AS
SELECT DISTINCT Rides_Tickets.ticket_ID AS scanned_ticket1
FROM Ride, Rides_Tickets
WHERE Ride.ride_ID = Rides_Tickets.ride_ID AND
```

```
year_started = '2006';

SELECT COUNT(DISTINCT scanned_ticket1)
FROM newride06;

CREATE VIEW newride07 AS
SELECT DISTINCT Rides_Tickets.ticket_ID AS scanned_ticket2
FROM Ride, Rides_Tickets
WHERE Ride.ride_ID = Rides_Tickets.ride_ID AND
year_started = '2007';

SELECT COUNT(DISTINCT scanned_ticket2)
FROM newride07;
```

Output Screenshot:



				ride_name	year_started
☐	Edit	Copy	Delete	Air Papillon	2005
☐	Edit	Copy	Delete	Goliath	2006
☐	Edit	Copy	Delete	Le Galopant	2007

```
COUNT(DISTINCT scanned_ticket)
92

COUNT(DISTINCT scanned_ticket1)
94

COUNT(DISTINCT scanned_ticket2)
75
```

From the above outputs, we can see that out of the three rides that started operating since 2000, the most popular is the "Goliath", which had 94 distinct ticket IDs associated with it.

Query 6:

Objective: Determine if customers with annual passes prefer to take the old or the new rides at La Ronde.

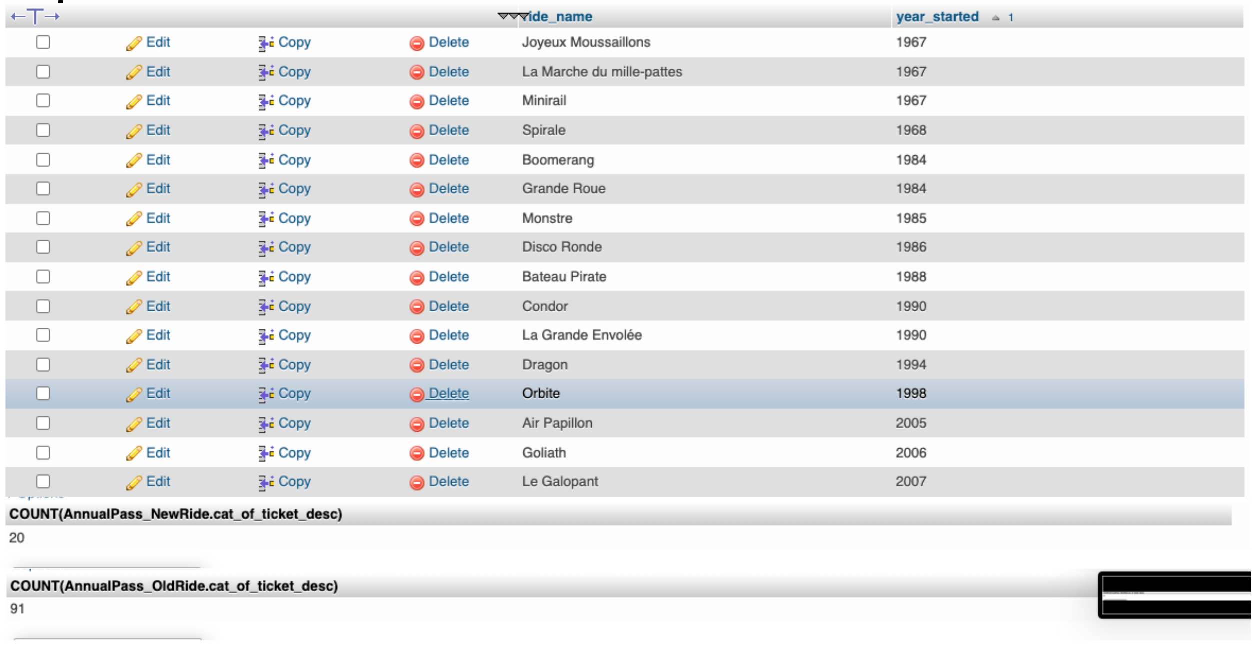
Assumptions: The year_started column in the Ride table indicates the year that a ride started operating, and if there is no year included in this column for a particular ride, then the year that this ride started operating is unknown. Old rides are the ones that started operating in the least recent year in the year_started column, and new rides are the ones that started operating in the most recent year in the year_started column.

Code:

```
SELECT ride_name, year_started
FROM Ride
WHERE year_started != 'null'
ORDER BY year_started;

CREATE VIEW AnnualPass_NewRide AS
SELECT Ticket.cat_of_ticket_desc, Ride.year_started, Ride.ride_name
FROM Ride, Rides_Tickets, Ticket
WHERE Ride.ride_ID = Rides_Tickets.ride_ID AND
```

```
Rides_Tickets.ticket_ID = Ticket.ticket_ID AND
Ride.year_started = "2007" AND
Ticket.cat_of_ticket_desc = "Annual Pass";

CREATE VIEW AnnualPass_OldRide AS
SELECT Ticket.cat_of_ticket_desc, Ride.year_started, Ride.ride_name
FROM Ride, Rides_Tickets, Ticket
WHERE Ride.ride_ID = Rides_Tickets.ride_ID AND
Rides_Tickets.ticket_ID = Ticket.ticket_ID AND
Ride.year_started = "1967" AND
Ticket.cat_of_ticket_desc = "Annual Pass";

SELECT COUNT(AnnualPass_NewRide.cat_of_ticket_desc)
FROM AnnualPass_NewRide;

SELECT COUNT(AnnualPass_OldRide.cat_of_ticket_desc)
FROM AnnualPass_OldRide;
```

Output Screenshot:



				ride_name	year_started
☐	Edit	Copy	Delete	Joyeux Moussaillons	1967
☐	Edit	Copy	Delete	La Marche du mille-pattes	1967
☐	Edit	Copy	Delete	Minirail	1967
☐	Edit	Copy	Delete	Spirale	1968
☐	Edit	Copy	Delete	Boomerang	1984
☐	Edit	Copy	Delete	Grande Roue	1984
☐	Edit	Copy	Delete	Monstre	1985
☐	Edit	Copy	Delete	Disco Ronde	1986
☐	Edit	Copy	Delete	Bateau Pirate	1988
☐	Edit	Copy	Delete	Condor	1990
☐	Edit	Copy	Delete	La Grande Envolée	1990
☐	Edit	Copy	Delete	Dragon	1994
☐	Edit	Copy	Delete	Orbite	1998
☐	Edit	Copy	Delete	Air Papillon	2005
☐	Edit	Copy	Delete	Goliath	2006
☐	Edit	Copy	Delete	Le Galopant	2007

```
COUNT(AnnualPass_NewRide.cat_of_ticket_desc)
20

COUNT(AnnualPass_OldRide.cat_of_ticket_desc)
91
```

[redacted]

From the above outputs we can see that the newest rides to begin operating at La Ronde is La Galopant, and the oldest rides to begin operating are Joyeux Moussaillons, La Marche du mille-pattes, and Minirail. We can then see that customers with the annual pass prefer the older rides, over the new ride since 20 annual pass tickets were scanned at the new ride, and 91 annual pass tickets were scanned at the older rides. Granted, we cannot say that customers prefer all three of the older rides to the new ride, but there must be a least one older ride which customers prefer to the newer ride.

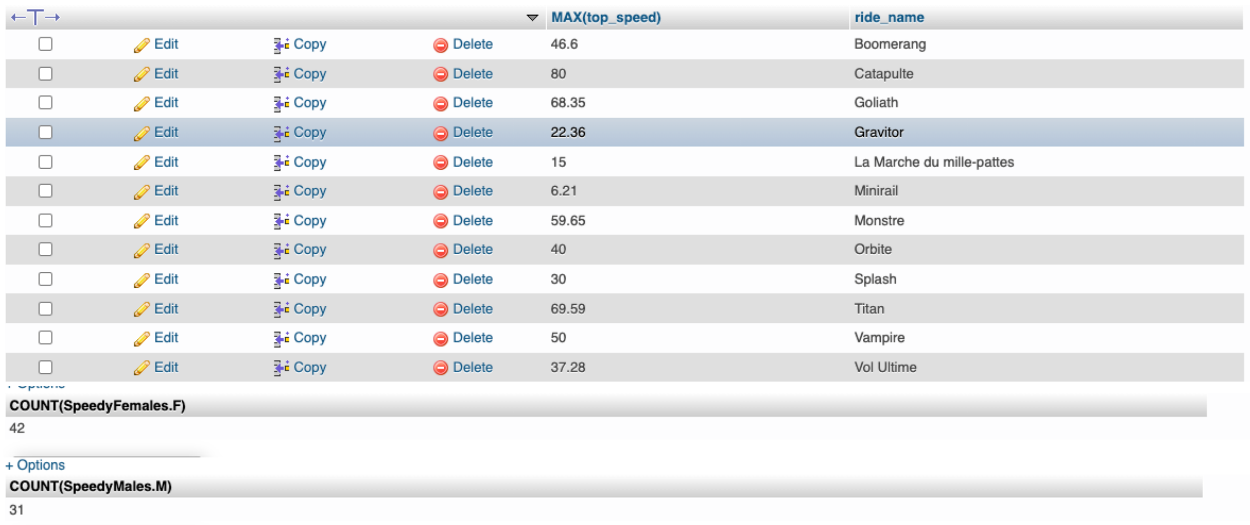
Query 7:

Objective: Do female or male customers like fast rides more? Determine this by counting the number of female and male customers who have taken the fastest ride at La Ronde.
Assumption: the fastest ride is the one with the highest known top speed.

Code:

```
SELECT MAX(top_speed), ride_name
FROM Ride
WHERE top_speed != 'null'
GROUP BY ride_name

CREATE VIEW SpeedyFemales AS
SELECT DISTINCT Customer.customer_ID as F, ride_name, top_speed, Customer.gender
FROM Ride, Rides_Tickets, Ticket, Customer
WHERE Ride.ride_ID = Rides_Tickets.ride_ID AND
Rides_Tickets.ticket_ID = Ticket.ticket_ID AND
Ticket.customer_ID = Customer.customer_ID AND
top_speed = '80'AND
Customer.gender = "Female";

CREATE VIEW SpeedyMales AS
SELECT DISTINCT Customer.customer_ID as M, ride_name, top_speed, Customer.gender
FROM Ride, Rides_Tickets, Ticket, Customer
WHERE Ride.ride_ID = Rides_Tickets.ride_ID AND
Rides_Tickets.ticket_ID = Ticket.ticket_ID AND
Ticket.customer_ID = Customer.customer_ID AND
top_speed = '80' AND
Customer.gender = "Male";

SELECT COUNT(SpeedyFemales.F)
FROM SpeedyFemales;

SELECT COUNT(SpeedyMales.M)
FROM SpeedyMales;
```

Output Screenshot:



MAX(top_speed)	ride_name
46.6	Boomerang
80	Catapulte
68.35	Goliath
22.36	Gravitor
15	La Marche du mille-pattes
6.21	Minirail
59.65	Monstre
40	Orbite
30	Splash
69.59	Titan
50	Vampire
37.28	Vol Ultime

COUNT(SpeedyFemales.F)
42

+ Options

COUNT(SpeedyMales.M)
31

From the outputs we can see that the fastest ride is Catapulte, with a top speed of 80 mph. Then, we can see that female customers like fast rides more than male customers do because more females (42) than males (31) have taken Catapulte.

Query 8:

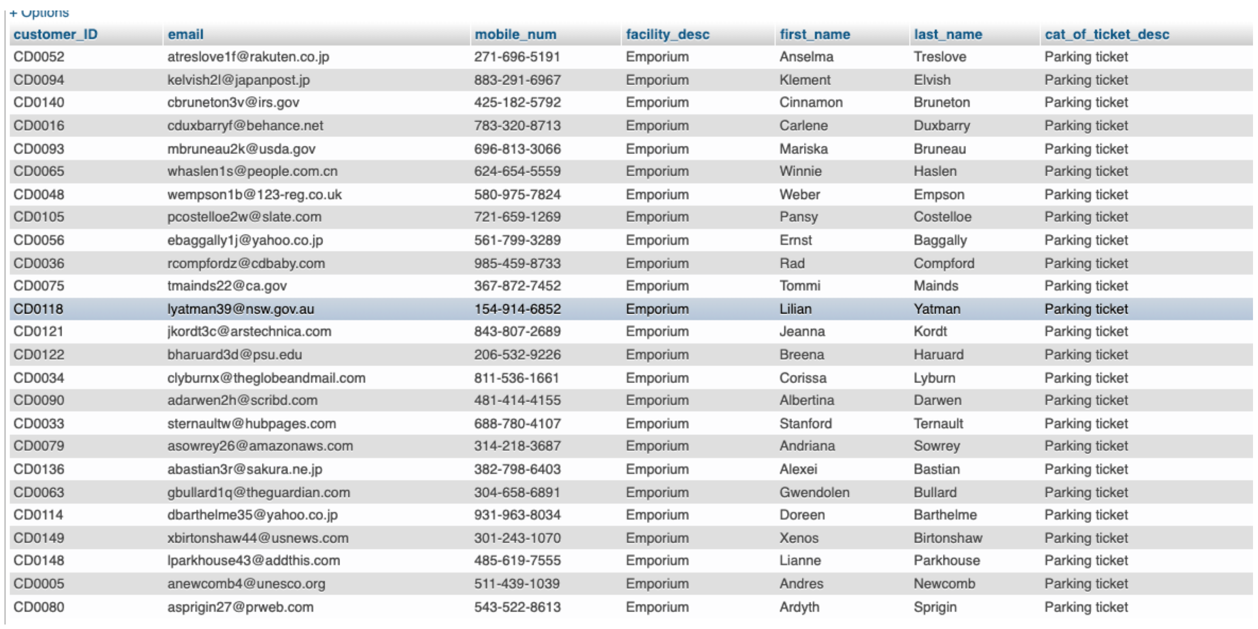
Objective: The Emporium facility would like to begin offering parking validation to customers who shop there while they are at La Ronde. To launch this parking validation program, Emporium's management would like to contact all the customers who have shopped at Emporium and purchased parking so that they can send out an advertisement telling these customers that they can validate their parking if they park and shop at Emporium again in the future.

Show all the customers (name and contact information) who have paid for parking and then shopped at Emporium.

Code:

```
SELECT DISTINCT Customer.customer_ID, email, mobile_num, facility_desc,
Customer.first_name, Customer.last_name, Ticket.cat_of_ticket_desc
FROM Facility, Facilities_Tickets, Ticket, Customer
WHERE Facility.facility_ID = Facilities_Tickets.facility_ID AND
Facilities_Tickets.ticket_ID = Ticket.ticket_ID AND
Ticket.customer_ID = Customer.customer_ID AND
facility_desc = "Emporium" AND
Ticket.cat_of_ticket_desc = "Parking ticket";
```

Output Screenshot:



+ Options

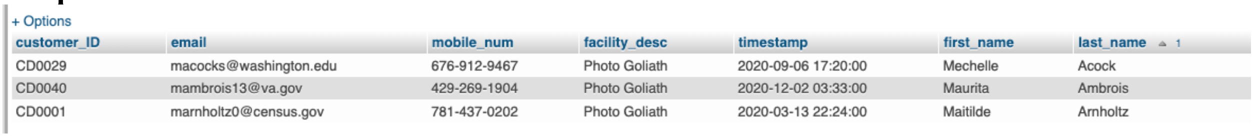
customer_ID	email	mobile_num	facility_desc	first_name	last_name	cat_of_ticket_desc
CD0052	atreslove1f@rakuten.co.jp	271-696-5191	Emporium	Anselma	Treslove	Parking ticket
CD0094	kelvish2l@japanpost.jp	883-291-6967	Emporium	Klement	Elvish	Parking ticket
CD0140	cbruneton3v@irs.gov	425-182-5792	Emporium	Cinnamon	Bruneton	Parking ticket
CD0016	cduxbarryf@behance.net	783-320-8713	Emporium	Carlene	Duxbarry	Parking ticket
CD0093	mbruneau2k@usda.gov	696-813-3066	Emporium	Mariska	Bruneau	Parking ticket
CD0065	whaslen1s@people.com.cn	624-654-5559	Emporium	Winnie	Haslen	Parking ticket
CD0048	wempson1b@123-reg.co.uk	580-975-7824	Emporium	Weber	Empson	Parking ticket
CD0105	pcostelloe2w@slate.com	721-659-1269	Emporium	Pansy	Costelloe	Parking ticket
CD0056	ebaggally1j@yahoo.co.jp	561-799-3289	Emporium	Ernst	Baggally	Parking ticket
CD0036	rcompfordz@cdbaby.com	985-459-8733	Emporium	Rad	Compford	Parking ticket
CD0075	tmainds22@ca.gov	367-872-7452	Emporium	Tommi	Mainds	Parking ticket
CD0118	lyatman39@nsw.gov.au	154-914-6852	Emporium	Lilian	Yatman	Parking ticket
CD0121	jkordt3c@arstechnica.com	843-807-2689	Emporium	Jeanna	Kordt	Parking ticket
CD0122	bharuard3d@psu.edu	206-532-9226	Emporium	Breena	Haruard	Parking ticket
CD0034	clyburnx@theglobeandmail.com	811-536-1661	Emporium	Corissa	Lyburn	Parking ticket
CD0090	adarwen2h@scribd.com	481-414-4155	Emporium	Albertina	Darwen	Parking ticket
CD0033	sternaultw@hubpages.com	688-780-4107	Emporium	Stanford	Ternault	Parking ticket
CD0079	asowrey26@amazonaws.com	314-218-3687	Emporium	Andriana	Sowrey	Parking ticket
CD0136	abastian3r@sakura.ne.jp	382-798-6403	Emporium	Alexei	Bastian	Parking ticket
CD0063	gbullard1q@theguardian.com	304-658-6891	Emporium	Gwendolen	Bullard	Parking ticket
CD0114	dbarthelme35@yahoo.co.jp	931-963-8034	Emporium	Doreen	Barthelme	Parking ticket
CD0149	xbirtonshaw44@usnews.com	301-243-1070	Emporium	Xenos	Birtonshaw	Parking ticket
CD0148	lparkhouse43@addthis.com	485-619-7555	Emporium	Lianne	Parkhouse	Parking ticket
CD0005	anewcomb4@unesco.org	511-439-1039	Emporium	Andres	Newcomb	Parking ticket
CD0080	asprigin27@prweb.com	543-522-8613	Emporium	Ardyth	Sprigin	Parking ticket

Query 9:
Objective: The management of the Photo Goliath facility is cleaning out its lost and found from last year (2020), and there is a briefcase with the initials M.A. on it. Show the manager the contact information and full name of all the customer(s) who this briefcase *could* belong to.

Code:

```
SELECT Customer.customer_ID, email, mobile_num, facility_desc, Facilities_Tickets.timestamp,
Customer.first_name, Customer.last_name
FROM Facility, Facilities_Tickets, Ticket, Customer
WHERE Facility.facility_ID = Facilities_Tickets.facility_ID AND
Facilities_Tickets.ticket_ID = Ticket.ticket_ID AND
Ticket.customer_ID = Customer.customer_ID AND
facility_desc = "Photo Goliath" AND
Facilities_Tickets.timestamp > "2019-12-31" AND
first_name LIKE "M%" AND
last_name LIKE "A%"
ORDER BY last_name;
```

Output Screenshot:



+ Options

customer_ID	email	mobile_num	facility_desc	timestamp	first_name	last_name ▲ 1
CD0029	macocks@washington.edu	676-912-9467	Photo Goliath	2020-09-06 17:20:00	Mechelle	Acock
CD0040	mambrois13@va.gov	429-269-1904	Photo Goliath	2020-12-02 03:33:00	Maurita	Ambrois
CD0001	marnholtz0@census.gov	781-437-0202	Photo Goliath	2020-03-13 22:24:00	Maitilde	Arnholtz

The briefcase could only belong to the three customers shown in the above screenshot.

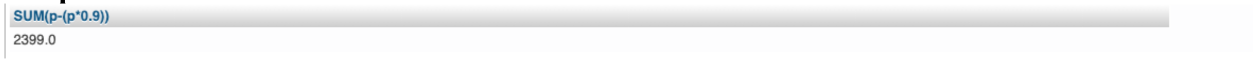
Query 10:
Objective: La Ronde’s management is considering a 10% discount for all customers who are associated with American universities to encourage them to travel to Montreal to visit the park. To

help management evaluate the possibility of staring such a discount program, show them the total revenue (in dollars) that would have been lost if every past customer who paid for a ticket and used a ".edu" email address was given 10% off.
Assumption: An ".edu" email address indicates that a customer is associated (student, staff, faculty, etc.) with and American university.

Code:

```
CREATE VIEW edu AS
SELECT Customer.customer_ID, Customer.first_name, Customer.last_name, Customer.email,
Ticket.price as p
FROM Ticket, Customer
WHERE Customer.customer_ID = Ticket.customer_ID AND
Customer.email LIKE "%edu";

SELECT SUM(p-(p*0.9))
FROM edu;
```

Output Screenshot:



```
SUM(p-(p*0.9))
2399.0
```

From the above output, we can see that the total revenue (in dollars) that would have been lost is $2399, if every past customer who paid for a ticket and used an ".edu" email address was given 10% off.