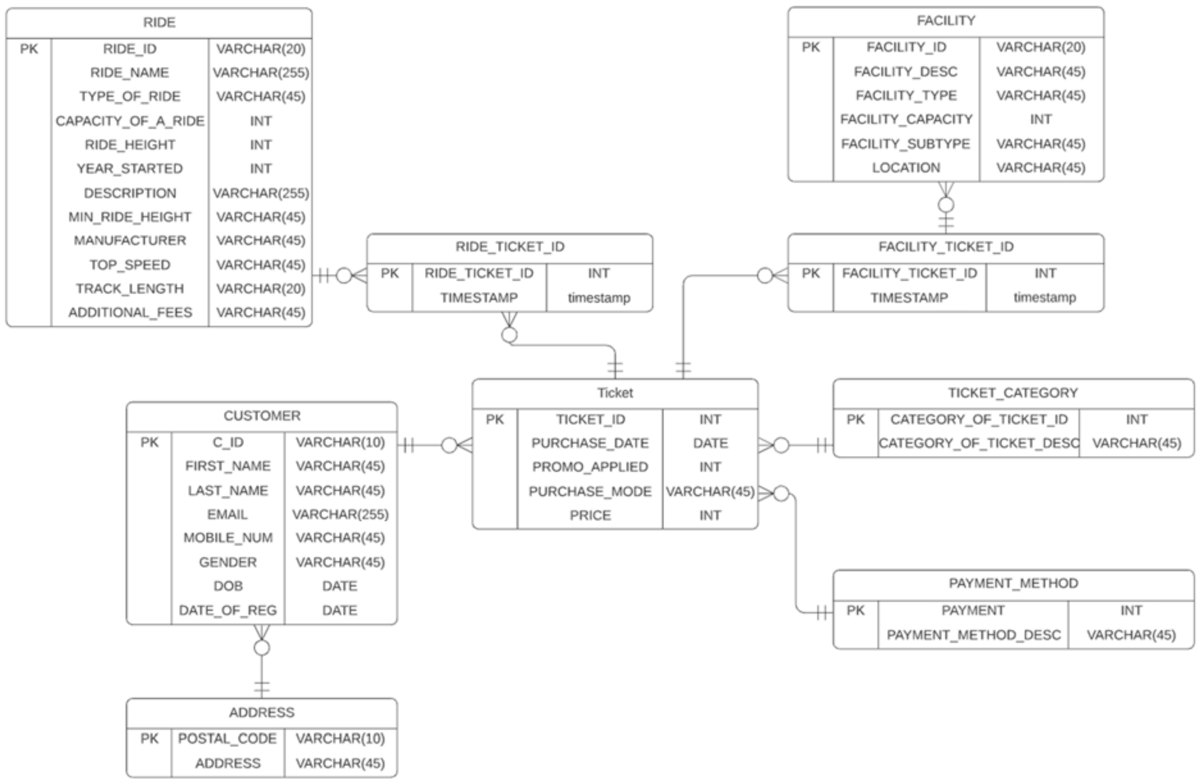
INSY 661 INDIVIDUAL PROJECTS REPORT

Yichen Wang

260761601

(20%) ERD: You have to analyse the data stored in the files and then create an ERD (identify what should be an entity, what attribute each entity should have, and how each entity should be related) to meet the given business requirements.



The tables given are already in the first normal form as there are no multivalued attributes. To reach the third normal form, we need to get rid off the transitive dependencies, hence ticket_category and payment_method are removed from ticket and address is removed from customer.

2) **(10%) RELATIONAL MODEL**: You should then create a relational model based on your ERD.

Use DDL to create tables in a MySQL database, as per your relational model. Link related tables together with foreign key constraints.

RIDE (RIDE_ID, RIDE_NAME, TYPE_OF_RIDE, CAPACITY_OF_A_RIDE, RIDE_HEIGHT, YEAR_STARTED, DESCRIPTION, MIN_RIDE_HEIGHT, MANUFACTURER, TOP_SPEED, TRACK_LENGTH, ADDITIONAL_FEES)

FACILITY (FACILITY_ID, FACILITY_DESC, FACILITY_TYPE, FACILITY_CAPACITY, FACILITY_SUBTYPE, LOCATION)

TICKET_CATEGORY (CATEGORY_OF_TICKET_ID, CATEGORY_OF_TICKET_DESC)

PAYMENT_METHOD (PAYMENT, PAYMENT_METHOD_DESC)

ADDRESS (POSTAL_CODE, ADDRESS)

CUSTOMER (C_ID, FIRST_NAME, LAST_NAME, EMAIL, MOBILE_NUM, GENDER, DOB, DATE_OF_REG, POSTAL_CODE)

TICKET (TICKET_ID, PURCHASE_DATE, PROMO_APPLIED, PURCHASE_MODE, PRICE, CATEGORY_OF_TICKET_ID, PAYMENT, C_ID)

FACILITY_TICKET_ID (FACILITY_TICKET_ID, TIMESTAMP, TICKET_ID, FACILITY_ID)

RIDE_TICKET_ID (RIDE_TICKET_ID, TIMESTAMP, TICKET_ID, RIDE_ID)

3) **(10%) POPULATE TABLES**: Use the data provided in the given CSV files to populate the tables that you have created in your database. NOTE: You should import all the files as tables in a MySQL database. These will act as temporary tables (youo can delete them later) that will be used to feed data (using INSERT) into tables that you have created as per your relational schema.

See the given sql files

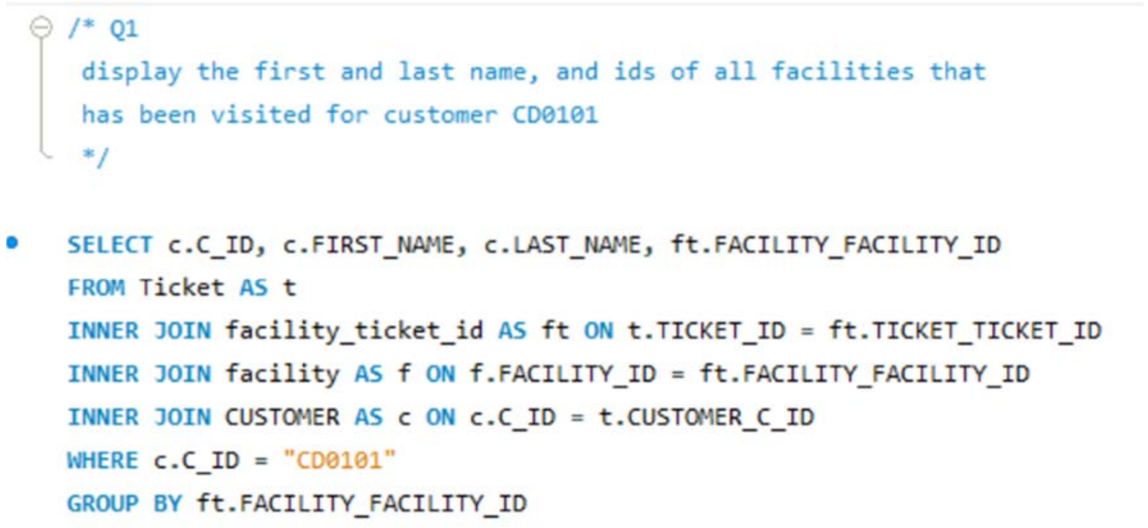
Since all the data has been cleaned using python before importing into the database, all the data insertion process will not be shown as only Import Wizard of the MySQL workbench is used.

4) **(20%) Queries: Prepare 10 SQL queries (Complex) that will give interesting insights** (e.g., derived value, aggregate functions, etc.).

Q1. To check which facilities are popular within la Ronde, we will randomly choose a customer and check his/her visit history. Display the first and last name, and ids of all facilities that has been visited for customer CD0101.

Assumption: we will check the history for all the dates and only unique IDs will be recorded.

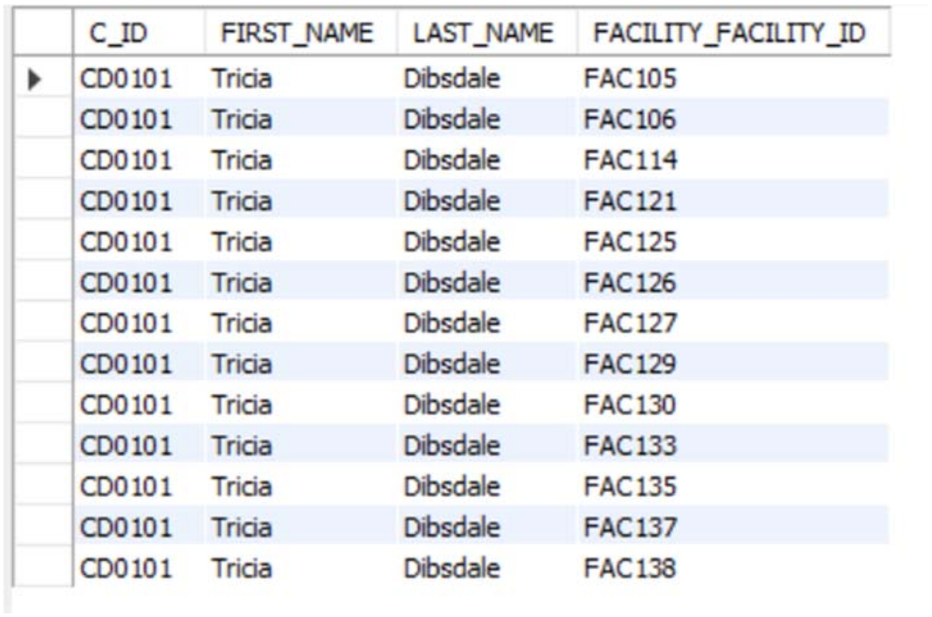
Codes:



```
/* Q1
 display the first and last name, and ids of all facilities that
 has been visited for customer CD0101
 */

SELECT c.C_ID, c.FIRST_NAME, c.LAST_NAME, ft.FACILITY_FACILITY_ID
FROM Ticket AS t
INNER JOIN facility_ticket_id AS ft ON t.TICKET_ID = ft.TICKET_TICKET_ID
INNER JOIN facility AS f ON f.FACILITY_ID = ft.FACILITY_FACILITY_ID
INNER JOIN CUSTOMER AS c ON c.C_ID = t.CUSTOMER_C_ID
WHERE c.C_ID = "CD0101"
GROUP BY ft.FACILITY_FACILITY_ID
```

Results:

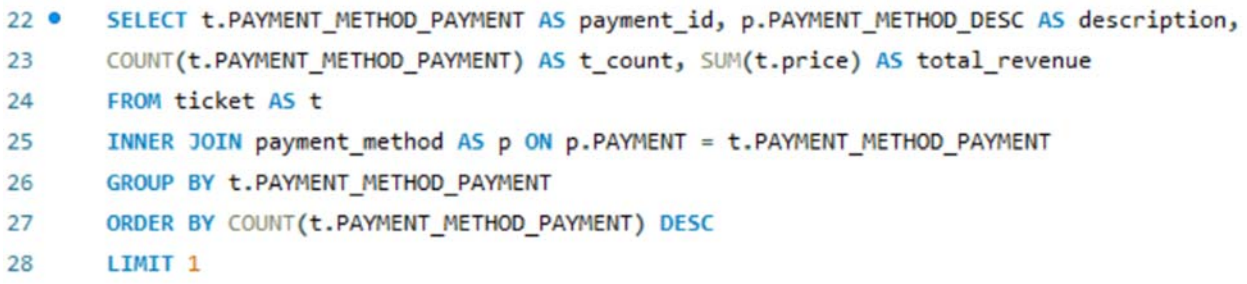


C_ID	FIRST_NAME	LAST_NAME	FACILITY_FACILITY_ID
CD0101	Tricia	Dibsdale	FAC105
CD0101	Tricia	Dibsdale	FAC106
CD0101	Tricia	Dibsdale	FAC114
CD0101	Tricia	Dibsdale	FAC121
CD0101	Tricia	Dibsdale	FAC125
CD0101	Tricia	Dibsdale	FAC126
CD0101	Tricia	Dibsdale	FAC127
CD0101	Tricia	Dibsdale	FAC129
CD0101	Tricia	Dibsdale	FAC130
CD0101	Tricia	Dibsdale	FAC133
CD0101	Tricia	Dibsdale	FAC135
CD0101	Tricia	Dibsdale	FAC137
CD0101	Tricia	Dibsdale	FAC138

Q2. The company will promote the most popular payment method if they have the related information. Help la Ronde improve the revenue by display the most popular method of payment based on its amount of sales among all the purchases, as well as its total revenue generated

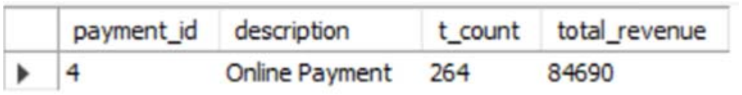
Assumption: revenue is calculated using sum of the product of ticket price and the number of tickets sold.

Codes:



```sql
SELECT t.PAYMENT_METHOD_PAYMENT AS payment_id, p.PAYMENT_METHOD_DESC AS description,
COUNT(t.PAYMENT_METHOD_PAYMENT) AS t_count, SUM(t.price) AS total_revenue
FROM ticket AS t
INNER JOIN payment_method AS p ON p.PAYMENT = t.PAYMENT_METHOD_PAYMENT
GROUP BY t.PAYMENT_METHOD_PAYMENT
ORDER BY COUNT(t.PAYMENT_METHOD_PAYMENT) DESC
LIMIT 1
```

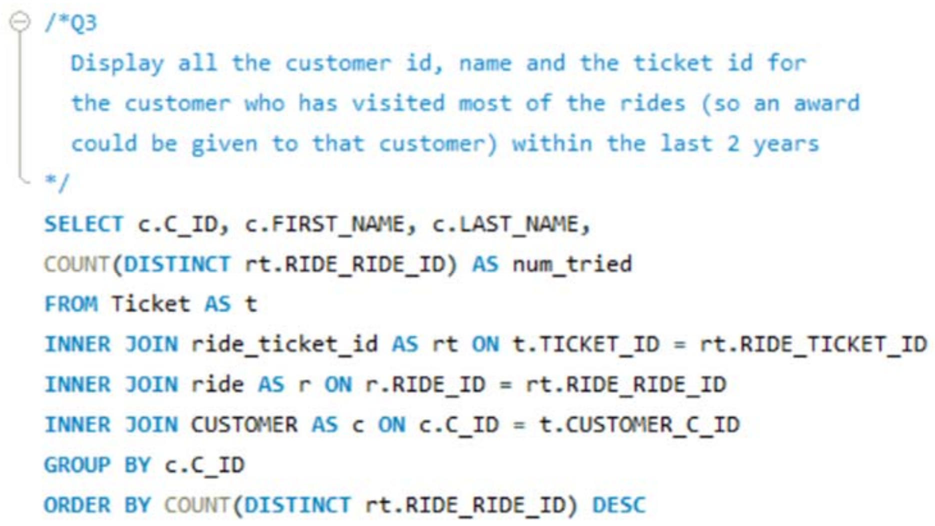
Results:



payment_id	description	t_count	total_revenue
4	Online Payment	264	84690

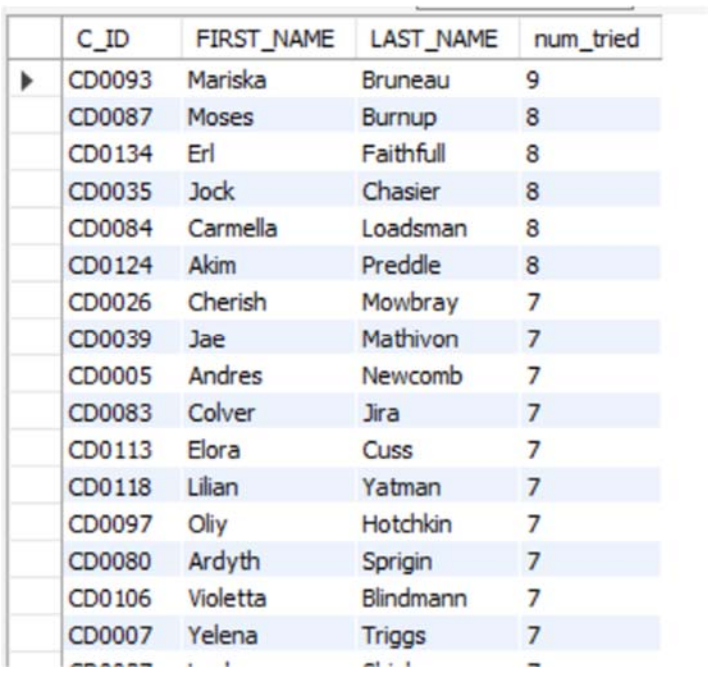
Q3: in order to reward loyal customers and promote their rides, Display all the customer id, name and the ticket id for the customer who has visited most of the rides (so an award could be given to that customer) within the last 2 years (in descending order)

Codes:



```sql
/*Q3
  Display all the customer id, name and the ticket id for
  the customer who has visited most of the rides (so an award
  could be given to that customer) within the last 2 years
*/
SELECT c.C_ID, c.FIRST_NAME, c.LAST_NAME,
COUNT(DISTINCT rt.RIDE_RIDE_ID) AS num_tried
FROM Ticket AS t
INNER JOIN ride_ticket_id AS rt ON t.TICKET_ID = rt.RIDE_TICKET_ID
INNER JOIN ride AS r ON r.RIDE_ID = rt.RIDE_RIDE_ID
INNER JOIN CUSTOMER AS c ON c.C_ID = t.CUSTOMER_C_ID
GROUP BY c.C_ID
ORDER BY COUNT(DISTINCT rt.RIDE_RIDE_ID) DESC
```

Results (partial):

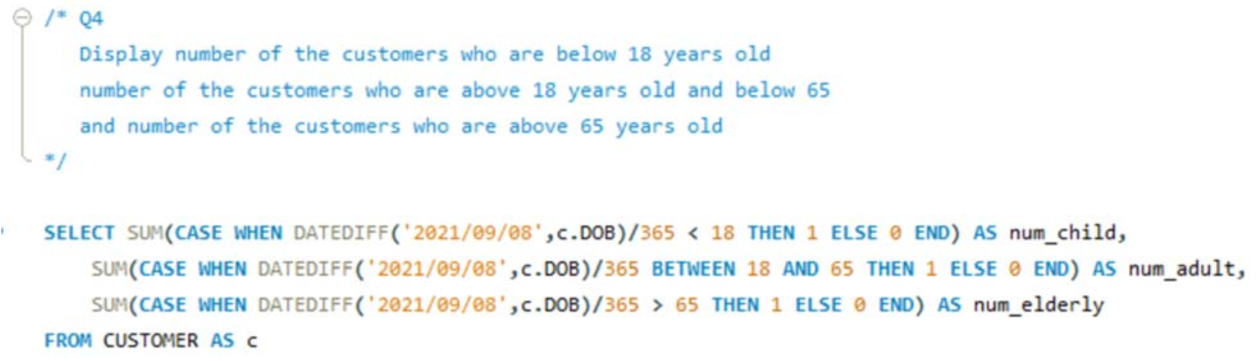


C_ID	FIRST_NAME	LAST_NAME	num_tried
CD0093	Mariska	Bruneau	9
CD0087	Moses	Burnup	8
CD0134	Erl	Faithfull	8
CD0035	Jock	Chasier	8
CD0084	Carmella	Loadsman	8
CD0124	Akim	Preddle	8
CD0026	Cherish	Mowbray	7
CD0039	Jae	Mathivon	7
CD0005	Andres	Newcomb	7
CD0083	Colver	Jira	7
CD0113	Elora	Cuss	7
CD0118	Lilian	Yatman	7
CD0097	Oliy	Hotchkin	7
CD0080	Ardyth	Sprigin	7
CD0106	Violetta	Blindmann	7
CD0007	Yelena	Triggs	7

Q4. To understand better the age structure of all the customers visiting la Ronde, display number of the customers who are below 18 years old number of the customers who are above 18 years old and below 65 and number of the customers who are above 65 years old

Assumption: compared with the date of 2021/09/08

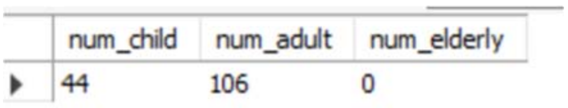
Codes:



```
/* Q4
   Display number of the customers who are below 18 years old
   number of the customers who are above 18 years old and below 65
   and number of the customers who are above 65 years old
*/

SELECT SUM(CASE WHEN DATEDIFF('2021/09/08',c.DOB)/365 < 18 THEN 1 ELSE 0 END) AS num_child,
       SUM(CASE WHEN DATEDIFF('2021/09/08',c.DOB)/365 BETWEEN 18 AND 65 THEN 1 ELSE 0 END) AS num_adult,
       SUM(CASE WHEN DATEDIFF('2021/09/08',c.DOB)/365 > 65 THEN 1 ELSE 0 END) AS num_elderly
FROM CUSTOMER AS c
```

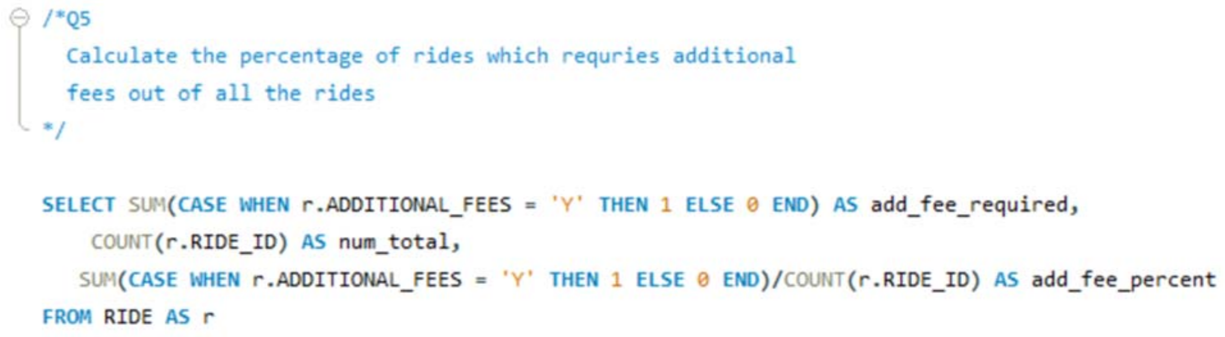
Results:



	num_child	num_adult	num_elderly
▶	44	106	0

Q5. To give all the customers a better idea of how many rides require additional fees, calculate the percentage of rides which requries additional fees out of all the rides.

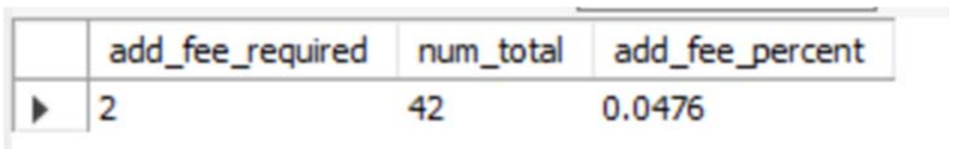
Codes:



```
/*Q5
  Calculate the percentage of rides which requries additional
  fees out of all the rides
*/

SELECT SUM(CASE WHEN r.ADDITIONAL_FEES = 'Y' THEN 1 ELSE 0 END) AS add_fee_required,
       COUNT(r.RIDE_ID) AS num_total,
      SUM(CASE WHEN r.ADDITIONAL_FEES = 'Y' THEN 1 ELSE 0 END)/COUNT(r.RIDE_ID) AS add_fee_percent
FROM RIDE AS r
```

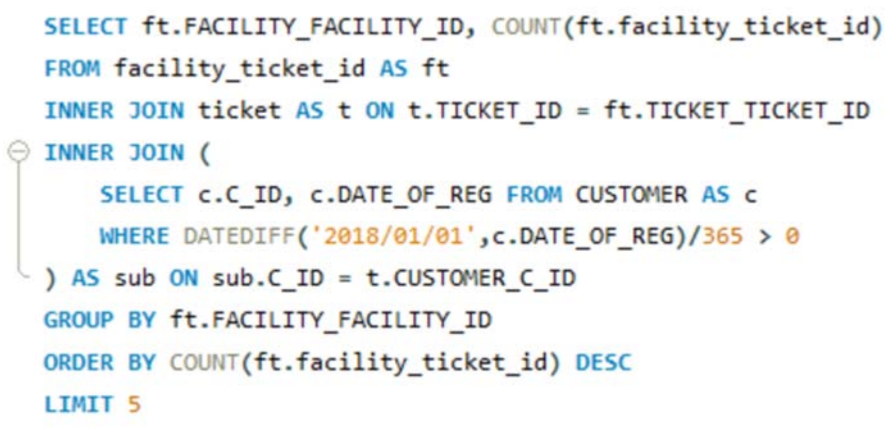
Results:



	add_fee_required	num_total	add_fee_percent
▶	2	42	0.0476

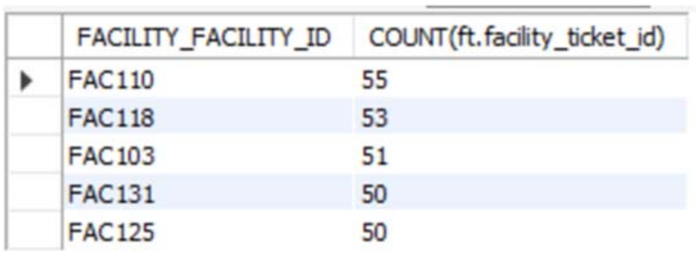
Q6. La ronde wants to check which facilities are remained popular. If customers who registered before 2018 are considered as early birds, find the facility which is in top 5 popularity among early birds.

Codes:



```
SELECT ft.FACILITY_FACILITY_ID, COUNT(ft.facility_ticket_id)
FROM facility_ticket_id AS ft
INNER JOIN ticket AS t ON t.TICKET_ID = ft.TICKET_TICKET_ID
INNER JOIN (
    SELECT c.C_ID, c.DATE_OF_REG FROM CUSTOMER AS c
    WHERE DATEDIFF('2018/01/01',c.DATE_OF_REG)/365 > 0
) AS sub ON sub.C_ID = t.CUSTOMER_C_ID
GROUP BY ft.FACILITY_FACILITY_ID
ORDER BY COUNT(ft.facility_ticket_id) DESC
LIMIT 5
```

Results:

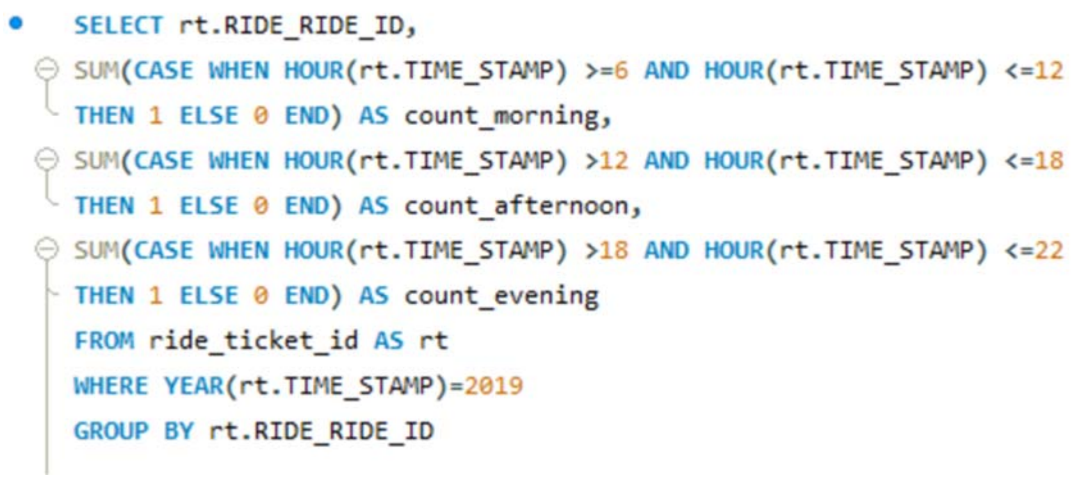


FACILITY_FACILITY_ID	COUNT(ft.facility_ticket_id)
FAC110	55
FAC118	53
FAC103	51
FAC131	50
FAC125	50

Q7. Apparent the number of visits to rides can be affected by the time. Count the number of visits of each rides in morning, afternoon and evening in 2019.

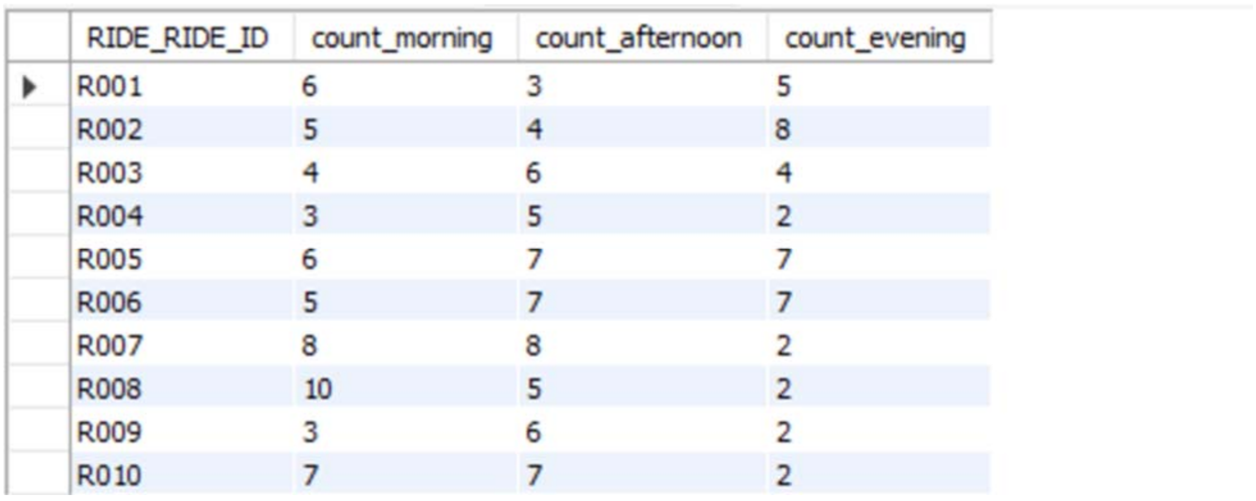
Assumption: morning is considered as from 6-12, afternoon is 12-18, evening is 18-22

Codes:



```
SELECT rt.RIDE_RIDE_ID,
SUM(CASE WHEN HOUR(rt.TIME_STAMP) >=6 AND HOUR(rt.TIME_STAMP) <=12
THEN 1 ELSE 0 END) AS count_morning,
SUM(CASE WHEN HOUR(rt.TIME_STAMP) >12 AND HOUR(rt.TIME_STAMP) <=18
THEN 1 ELSE 0 END) AS count_afternoon,
SUM(CASE WHEN HOUR(rt.TIME_STAMP) >18 AND HOUR(rt.TIME_STAMP) <=22
THEN 1 ELSE 0 END) AS count_evening
FROM ride_ticket_id AS rt
WHERE YEAR(rt.TIME_STAMP)=2019
GROUP BY rt.RIDE_RIDE_ID
```

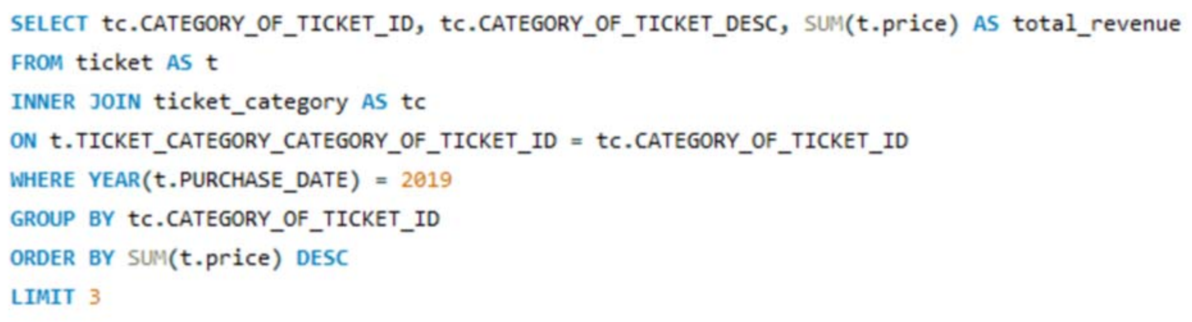
Results (partial):



RIDE_RIDE_ID	count_morning	count_afternoon	count_evening
R001	6	3	5
R002	5	4	8
R003	4	6	4
R004	3	5	2
R005	6	7	7
R006	5	7	7
R007	8	8	2
R008	10	5	2
R009	3	6	2
R010	7	7	2

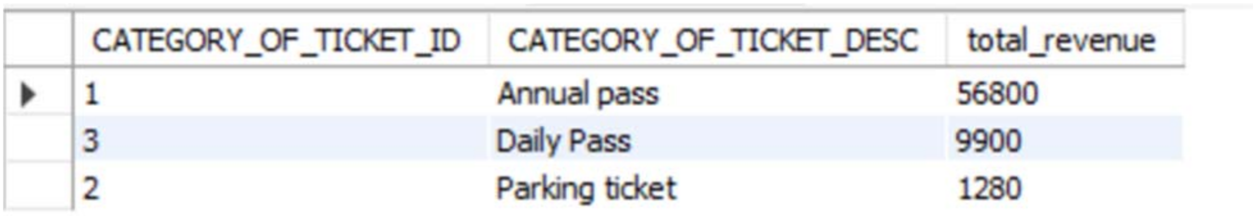
Q8: there are many different types of tickets. Check which category of ticket generates the most amount of revenue in DESC order in 2019.

Codes:



```
SELECT tc.CATEGORY_OF_TICKET_ID, tc.CATEGORY_OF_TICKET_DESC, SUM(t.price) AS total_revenue
FROM ticket AS t
INNER JOIN ticket_category AS tc
ON t.TICKET_CATEGORY_CATEGORY_OF_TICKET_ID = tc.CATEGORY_OF_TICKET_ID
WHERE YEAR(t.PURCHASE_DATE) = 2019
GROUP BY tc.CATEGORY_OF_TICKET_ID
ORDER BY SUM(t.price) DESC
LIMIT 3
```

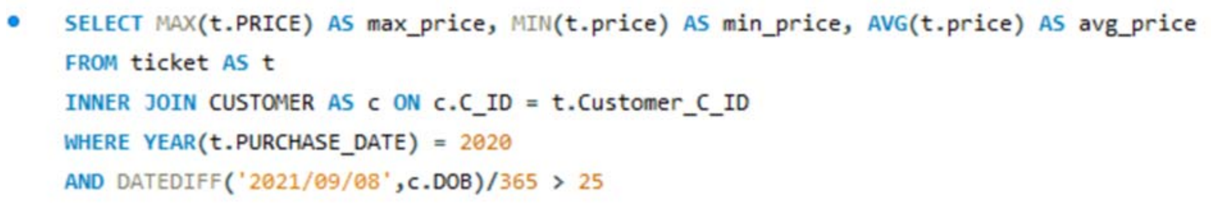
Results:



CATEGORY_OF_TICKET_ID	CATEGORY_OF_TICKET_DESC	total_revenue
1	Annual pass	56800
3	Daily Pass	9900
2	Parking ticket	1280

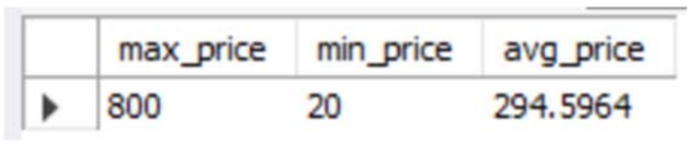
Q9: To see how well our promo codes are doing, check the highest price, lowest price and the average price paid by the consumer in year 2020 who are under the age of 25.

Codes:



```
SELECT MAX(t.PRICE) AS max_price, MIN(t.price) AS min_price, AVG(t.price) AS avg_price
FROM ticket AS t
INNER JOIN CUSTOMER AS c ON c.C_ID = t.Customer_C_ID
WHERE YEAR(t.PURCHASE_DATE) = 2020
AND DATEDIFF('2021/09/08',c.DOB)/365 > 25
```

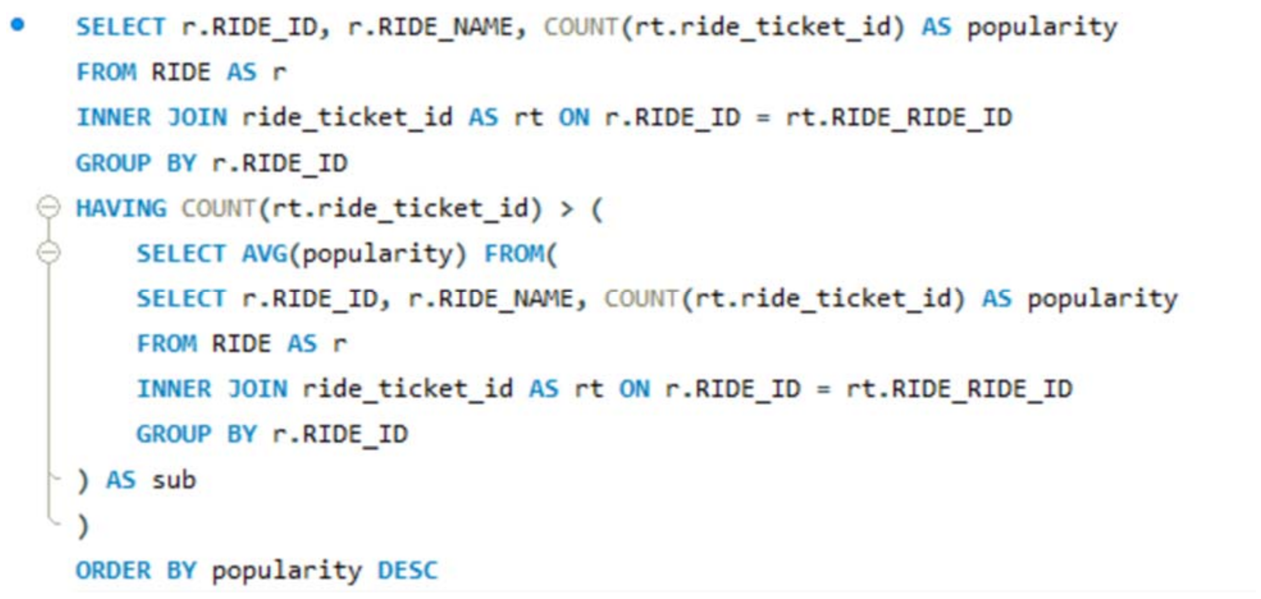
Results:



	max_price	min_price	avg_price
▶	800	20	294.5964

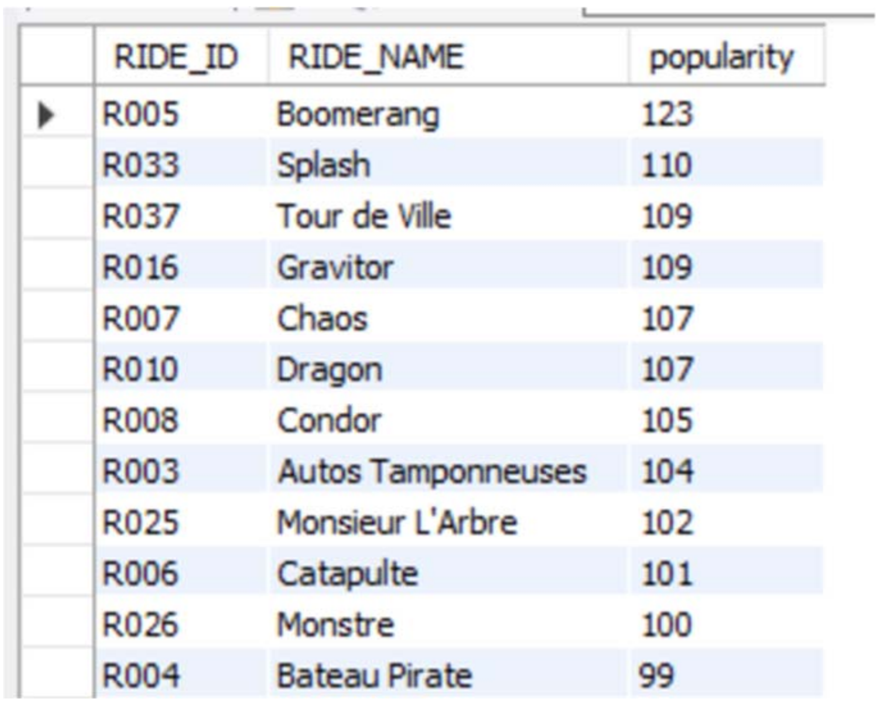
Q10: List down all the rides whose popularity based on times of visits is above average over the past few years.

Codes:



```
SELECT r.RIDE_ID, r.RIDE_NAME, COUNT(rt.ride_ticket_id) AS popularity
FROM RIDE AS r
INNER JOIN ride_ticket_id AS rt ON r.RIDE_ID = rt.RIDE_RIDE_ID
GROUP BY r.RIDE_ID
HAVING COUNT(rt.ride_ticket_id) > (
    SELECT AVG(popularity) FROM(
    SELECT r.RIDE_ID, r.RIDE_NAME, COUNT(rt.ride_ticket_id) AS popularity
    FROM RIDE AS r
    INNER JOIN ride_ticket_id AS rt ON r.RIDE_ID = rt.RIDE_RIDE_ID
    GROUP BY r.RIDE_ID
) AS sub
)
ORDER BY popularity DESC
```

Results (partial):



	RIDE_ID	RIDE_NAME	popularity
▶	R005	Boomerang	123
	R033	Splash	110
	R037	Tour de Ville	109
	R016	Gravitor	109
	R007	Chaos	107
	R010	Dragon	107
	R008	Condor	105
	R003	Autos Tamponneuses	104
	R025	Monsieur L'Arbre	102
	R006	Catapulte	101
	R026	Monstre	100
	R004	Bateau Pirate	99