Database Design PART I

FOR LA RONDE PARK, MONTREAL

Xintong Li |INSY 661| 2021-9-10

A Brief Introduction for this project

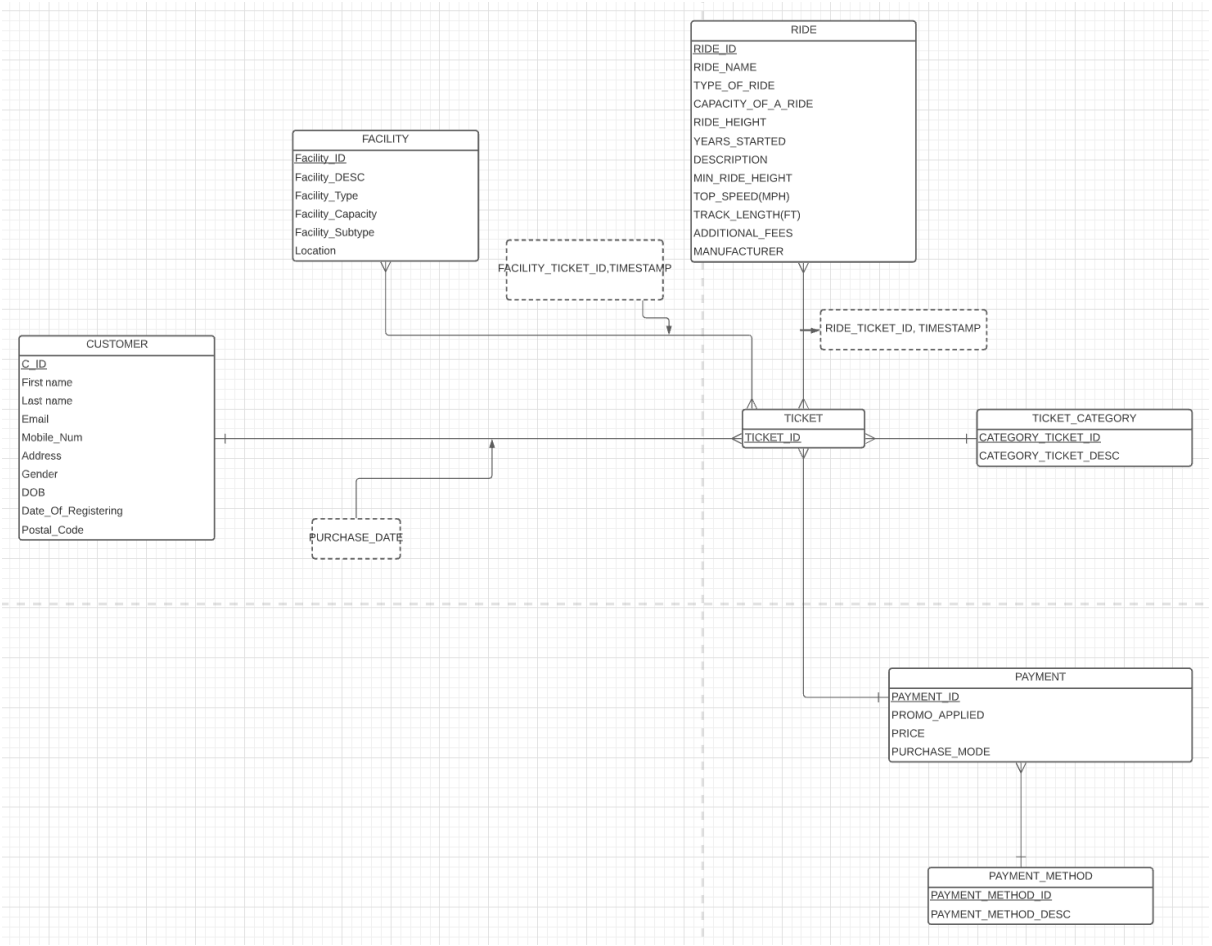
In this project, I will show how to clean and process the data crawled by the website and the data provided by the company/amusement park, how to build a ERD using them and how to implement the ERD to a real-world database from scratch.

This report is for the first part of the project, while in the second part I will also introduce covid-19 data to analyze the impact of the epidemic on a real economy like amusement parks.

1. ERD

   After doing normalization, I removed all the multi-value attributes, partial dependencies and transitive dependencies, and I got the ERD below:

   - The dotted boxes represent attributes for relationships, the arrow on it shows to which relationship it belongs to.
   - Assume that one can purchase many tickets in one payment, but one ticket can only belong to one payment.
   - The underline attributes represents primary key.
   - I did not include foreign key in ERD, but it will be shown in the logical model.



2. Logical Model

Note: the underline part means the primary key, the red circle represents foreign key.

- **Entities:**

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

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

RIDE(RIDE_ID, RIDE_NAME, TYPE_OF_RIDE, CAPACITY_OF_A_RIDE, RIDE_HEIGHT, YEARS_STARTED, DESCRIPTION, MIN_RIDE_HEIGHT, TOP_APEED(MPH), TRACK_LENGTH(FT), ADDITIONAL_FEES, MANUFACTURER)

TICKET_CATEGORY(CATEGORY_TICKET_ID, CATEGORY_TICKET_DESC)

PAYMENT(PAYMENT_ID, PAYMENT_METHOD_ID, PROMO_APPLIED, PRICE, PURCHASE_MODE)

PAYMENT_METHOD(PAYMENT_METHOD_ID, PAYMENT_METHOD_DESC)

TICKET(TICKET_ID, C_ID, CATEGORY_TICKET_ID, PAYMENT_METHOD_ID, PURCHASE_DATE)

- **Relationships:**

TICKET_FACILITY(FACILITY_ID, TICKET_ID, TIMESTAMP)

TICKET_RIDE(RIDE_ID, TICKET_ID, TIMESTAMP)

3. Queries

- DDL:

Note: You are not able to get all the raw data in your database just by simply running these queries, because the data given by the company is so messy that I must use python, excel and SQL to cleaned and preprocessed them for each table. After that, I also manually imported the cleaned csv files. So it is not possible that you could run these sql queries and get everything done in one second, please understand this.

But I will also submit the cleaned data, too. The cleaned data for part I is named with this pattern: '**XXX _without_external_data.csv**'. If you want to see my data, you could first run the create table part then import those csv files I submitted.

Besides these, I will also submit the python file which are used to clean the data incase if you would like to look them. The python file for part I is named with this pattern: '**Data_preprocessing_XXXX_without_external_data.py**'

```sql
CREATE TABLE CUSTOMER (
  C_ID VARCHAR(100) NOT NULL,
  FIRST_NAME VARCHAR(100),
  LAST_NAME VARCHAR(100),
  EMAIL VARCHAR(100),
  MOBILE_NUM VARCHAR(100),
  ADDRESS VARCHAR(100),
  GENDER VARCHAR(100),
  DOB DATE,
  DATE_OF_REG DATE,
  POSTAL_CODE VARCHAR(100),
  PRIMARY KEY (C_ID)
);

CREATE TABLE FACILITY (
  FACILITY_ID VARCHAR(100) NOT NULL,
  FACILITY_DESC VARCHAR(1000),
  FACILITY_TYPE VARCHAR(100),
  FACILITY_CAPACITY INT,
  FACILITY_SUBTYPE VARCHAR(1000),
  LOCATION VARCHAR(100),
  PRIMARY KEY (FACILITY_ID)
);

CREATE TABLE PAYMENT_METHOD (
  PAYMENT_METHOD_ID INT NOT NULL,
  PAYMENT_METHOD_DESC VARCHAR(100),
  PRIMARY KEY (PAYMENT_METHOD_ID)
);

CREATE TABLE PAYMENT (
  PAYMENT_ID INT NOT NULL,
  PAYMENT_METHOD_ID INT,
  PROMO_APPLIED INT(1),
  PRICE INT,
  PURCHASE_MODE VARCHAR(100),
  PRIMARY KEY (PAYMENT_ID),
  FOREIGN KEY (PAYMENT_METHOD_ID) REFERENCES PAYMENT_METHOD(PAYMENT_METHOD_ID)
);
```

```sql
CREATE TABLE TICKET_CATEGORY (
  CATEGORY_OF_TICKET_ID INT(1) NOT NULL,
  CATEGORY_OF_TICKET_DESC VARCHAR(100),
  PRIMARY KEY (CATEGORY_OF_TICKET_ID)
);

CREATE TABLE RIDE (
  RIDE_ID VARCHAR(100) NOT NULL,
  RIDE_NAME VARCHAR(100),
  TYPE_OF_RIDE VARCHAR(100),
  CAPACITY_OF_A_RIDE DOUBLE,
  RIDE_HEIGHT DOUBLE,
  YEARS_STARTED DOUBLE,
  DESCRIPTION VARCHAR(10000),
  MIN_RIDE_HEIGHT VARCHAR(1000),
   MANUFACTURER VARCHAR(100),
  `TOP_SPEED(MPH)` DOUBLE,
  `TRACK_LENGTH(FT)` DOUBLE,
  ADDITIONAL_FEES VARCHAR(1),
  PRIMARY KEY (RIDE_ID)
);

CREATE TABLE TICKET (
  TICKET_ID INT NOT NULL,
  C_ID VARCHAR(100),
  PAYMENT_ID INT NOT NULL,
  `CATEGORY_OF_TICKET_ID` INT(1),
    PURCHASE_DATE DATE,
  PRIMARY KEY (TICKET_ID, C_ID, PAYMENT_ID, `CATEGORY_OF_TICKET_ID`, PURCHASE_DATE),
  FOREIGN KEY (PAYMENT_ID) REFERENCES PAYMENT(PAYMENT_ID),
  FOREIGN KEY (C_ID) REFERENCES CUSTOMER(C_ID),
  FOREIGN KEY (`CATEGORY_OF_TICKET_ID`) REFERENCES `ticket_category`(`CATEGORY_OF_TICKET_ID`)
);
```

```sql
 CREATE TABLE TICKET_RIDE(
 RIDE_TICKET_ID INT NOT NULL,
 RIDE_ID VARCHAR(100),
 TICKET_ID INT,
 TIMESTAMP DATETIME,
 PRIMARY KEY (RIDE_TICKET_ID),
 FOREIGN KEY (TICKET_ID) REFERENCES TICKET(TICKET_ID),
 FOREIGN KEY (RIDE_ID) REFERENCES RIDE(RIDE_ID)
 );

 CREATE TABLE TICKET_FACILITY(
 FACILITY_TICKET_ID INT NOT NULL,
 TICKET_ID INT,
 FACILITY_ID VARCHAR(100),
 TIMESTAMP DATETIME,
 PRIMARY KEY (FACILITY_TICKET_ID),
 FOREIGN KEY (TICKET_ID) REFERENCES TICKET(TICKET_ID),
 FOREIGN KEY (FACILITY_ID) REFERENCES FACILITY(FACILITY_ID)
 );
```

```
/*the following is the query to manipulate RIDE table*/
/*I first use python to upload ride.csv to ride2 table in db, then use sql to fillna
then use sql to fill ride with ride2's data*/
/*I will also submit the python files I used to preprocess each table*/
UPDATE ride2
SET ride2.MIN_RIDE_HEIGHT= NULL
WHERE ride2.MIN_RIDE_HEIGHT='';

UPDATE ride2
SET ride2.CAPACITY_OF_A_RIDE= NULL
WHERE ride2.CAPACITY_OF_A_RIDE='';

UPDATE ride2
SET ride2.RIDE_HEIGHT= NULL
WHERE ride2.RIDE_HEIGHT='';


UPDATE ride2
SET ride2.YEARS_STARTED= NULL
WHERE ride2.YEARS_STARTED='';

UPDATE ride2
SET ride2.MANUFACTURER= NULL
WHERE ride2.MANUFACTURER='';

UPDATE ride2
SET ride2.`TOP_SPEED(MPH)`= NULL
WHERE ride2.`TOP_SPEED(MPH)`='';

UPDATE ride2
SET ride2.`TRACK_LENGTH(FT)`= NULL
WHERE ride2.`TRACK_LENGTH(FT)`='';
```

```
INSERT INTO
RIDE (RIDE_ID,`RIDE_NAME`,`TYPE_OF_RIDE`,`CAPACITY_OF_A_RIDE`,`RIDE_HEIGHT`,`YEARS_STARTED`,
`DESCRIPTION`,`MIN_RIDE_HEIGHT`,`MANUFACTURER`,`TOP_SPEED(MPH)`,`TRACK_LENGTH(FT)`,`ADDITIONAL_FEES`)
SELECT RIDE_ID,`RIDE_NAME` ,`TYPE_OF_RIDE` ,`CAPACITY_OF_A_RIDE`,`RIDE_HEIGHT`,`YEARS_STARTED` ,
`DESCRIPTION`,`MIN_RIDE_HEIGHT`,`MANUFACTURER`,`TOP_SPEED(MPH)`,`TRACK_LENGTH(FT)`,`ADDITIONAL_FEES`
FROM ride2;
```

- ## Ten queries:

  NOTE: it is impossible to get these queries to run unless you already have the correct the tables and data in the database, please understand this. For information about how to get the correct data in the database, please refer to the previous DDL section. If you are still having trouble getting the data into your database, feel free to leave me a message, I will be very happy to help you import them and get the queries run on your computer. (if needed, please leave me a message using this number: 4389277966)

1. **find the customers' (CID, emails and ages) who bought most annual-pass tickets without promotion**

(Business motivation: to find the contact information of customers who spend the most money in a single transaction, so as to send them messages such as advertisements since they are the most "generous" customers)

- Answer:

```
select temp.C_ID, customer_with_age.EMAIL, customer_with_age.age
from (
select *
from(
select count(ticket.TICKET_ID) AS Number_of_invest, customer_with_age.C_ID
from ticket
inner join payment
on ticket.PAYMENT_ID = payment.PAYMENT_ID
inner join ticket_category
on ticket.CATEGORY_OF_TICKET_ID =
ticket_category.CATEGORY_OF_TICKET_ID
inner join (
SELECT *, YEAR(CURDATE()) - YEAR(DOB) AS age FROM customer
) as customer_with_age
on ticket.C_ID = customer_with_age.C_ID
where payment.PROMO_APPLIED = 0 AND ticket.CATEGORY_OF_TICKET_ID =
1
group by customer_with_age.C_ID
ORDER BY count(ticket.TICKET_ID) DESC


) as a
where a.Number_of_invest = (select MAX(Number_of_invest) from (
select count(ticket.TICKET_ID) AS Number_of_invest, customer_with_age.C_ID
from ticket
inner join payment
on ticket.PAYMENT_ID = payment.PAYMENT_ID
inner join ticket_category
on ticket.CATEGORY_OF_TICKET_ID =
ticket_category.CATEGORY_OF_TICKET_ID
inner join (
SELECT *, YEAR(CURDATE()) - YEAR(DOB) AS age FROM customer
) as customer_with_age
on ticket.C_ID = customer_with_age.C_ID
where payment.PROMO_APPLIED = 0 AND ticket.CATEGORY_OF_TICKET_ID =
1
group by customer_with_age.C_ID
ORDER BY count(ticket.TICKET_ID) DESC
```

```
)as b
)
)as temp
inner join customer on temp.C_ID = customer.C_ID
inner join (
SELECT *, YEAR(CURDATE()) - YEAR(DOB) AS age FROM customer
) as customer_with_age
on temp.C_ID = customer_with_age.C_ID
```

- Output:
- The 3 person below are the customers who bought most annual tickets(4 annual tickets) without promotion.
- We got their email now so we can send them adds/coupons/whatever we could make profits from them, we also can see that they are all middle-aged people, this might indicate some The relationships between a person's purchasing power and his/her age.

C_ID	EMAIL	age
CD0022	gwetherilll@cmu.edu	31
CD0084	cloadsman2b@bigcartel.com	44
CD0113	ecuss34@w3.org	33

2. **find the TOP 10 customer's who spent most money in la ronde during the period we recorded. By doing this, we could do User Layering, and give those people who spent more money a VIP role, when they come to the amusement park, we could provide them some special services to make them feel distinguished, for example, we could provide and allow them to ride a horse in the park.**

- Answer:

```
select a.C_ID, SUM(a.PRICE) AS TOTAL_SPENT
from(
select TICKET.C_ID, PAYMENT.PRICE
from ticket
inner join payment on ticket.PAYMENT_ID = payment.PAYMENT_ID
)as a
group by a.C_ID
order by SUM(a.PRICE) DESC
LIMIT 10
```

	C_ID	TOTAL_SPENT
▶	CD0047	5160
	CD0124	5030
	CD0005	4930
	CD0022	4650
	CD0103	4540
	CD0105	4320
	CD0084	4210
	CD0030	4080
	CD0026	4060
	CD0092	3990

3. **Among each facility type, find the most popular subtype, by doing this, we can know which kind of facility is more popular and increase the number of those kind of facilities.**

- Answer:

```
select DISTINCT b.FACILITY_TYPE, b.max_visit, c.FACILITY_SUBTYPE
from(
select MAX(Number_of_visit) as max_visit, a.FACILITY_TYPE
from (
select facility.FACILITY_TYPE, facility.FACILITY_SUBTYPE,
subtype_visit.Number_of_visit from facility
inner join(
select facility.FACILITY_SUBTYPE, SUM(FACILITY_TICKET_ID) AS
Number_of_visit
from facility
inner join ticket_facility on facility.FACILITY_ID = ticket_facility.FACILITY_ID
group by facility.FACILITY_SUBTYPE

)as subtype_visit
on facility.FACILITY_SUBTYPE = subtype_visit.FACILITY_SUBTYPE
)as a
group by a.FACILITY_TYPE
)as b
inner join (
```

```
select facility.FACILITY_TYPE, facility.FACILITY_SUBTYPE,
subtype_visit.Number_of_visit from facility
inner join(
select facility.FACILITY_SUBTYPE, SUM(FACILITY_TICKET_ID) AS
Number_of_visit
from facility
inner join ticket_facility on facility.FACILITY_ID = ticket_facility.FACILITY_ID
group by facility.FACILITY_SUBTYPE

)as subtype_visit
on facility.FACILITY_SUBTYPE = subtype_visit.FACILITY_SUBTYPE
)as c
on b.max_visit= c.Number_of_visit
```

- Output:

	FACILITY_TYPE	max_visit	FACILITY_SUBTYPE
▶	Events	168222	
	Shopping	547998	Appareal
	Dinning	1631193	Ice Cream and Sweets

4. **Among each facility type, find the facility which creates most profit(according to the payment of its ticket)**

- Answer:

```
SELECT SUM(b.PRICE)as PROFIT, b.FACILITY_ID, b.FACILITY_DESC
FROM (
select a.FACILITY_ID, a.FACILITY_DESC, a.FACILITY_TICKET_ID AS
PER_ENTRANCE, PAYMENT.PRICE
from ticket
inner join(
select facility.FACILITY_DESC, ticket_facility.FACILITY_ID,
ticket_facility.TICKET_ID , TICKET_FACILITY.FACILITY_TICKET_ID from facility
inner join ticket_facility on facility.FACILITY_ID = ticket_facility.FACILITY_ID
)as a
on a.TICKET_ID = ticket.TICKET_ID
INNER JOIN PAYMENT ON TICKET.PAYMENT_ID = PAYMENT.PAYMENT_ID
)AS b
GROUP BY b.FACILITY_ID
order  by PROFIT DESC
LIMIT 10
```

- Output:

	PROFIT	FACILITY_ID	FACILITY_DESC
▶	38190	FAC139	Popcorn & Cie
	37570	FAC108	Boutique Splash
	36530	FAC110	Fines Poutines Express
	36490	FAC128	Au Comptoir Frais
	35340	FAC126	Rolopan 1
	33960	FAC109	Yo! Chine
	33840	FAC120	Moozoo 1
	33460	FAC117	Restaurant Lafleur
	32970	FAC142	Bar Rouge
	32950	FAC129	Halte Gourmande - Au Bol

5. **Find the top 3 rides that was most popular among girls/females. By doing this, we could detect females' needs when they go to amusement parks.**

- Answer:

```
SELECT c.TOTAL_VISIT, RIDE.RIDE_ID, RIDE.TYPE_OF_RIDE,
RIDE.DESCRIPTION
FROM (
SELECT COUNT(VISIT_ID) AS TOTAL_VISIT, RIDE_ID
FROM(
SELECT a.RIDE_ID, a.TYPE_OF_RIDE, a.DESCRIPTION, TICKET.C_ID,
CUSTOMER.GENDER, a.RIDE_TICKET_ID AS VISIT_ID
FROM TICKET
INNER JOIN (
SELECT RIDE.RIDE_ID, RIDE.TYPE_OF_RIDE, RIDE.DESCRIPTION,
TICKET_RIDE.TICKET_ID, TICKET_RIDE.RIDE_TICKET_ID FROM RIDE
INNER JOIN TICKET_RIDE
ON RIDE.RIDE_ID = ticket_ride.RIDE_ID
)AS a
ON TICKET.TICKET_ID = a.TICKET_ID
inner join customer
on TICKET.C_ID = CUSTOMER.C_ID
```

```
WHERE CUSTOMER.GENDER = 'Female'
)AS b
GROUP BY b.RIDE_ID
)AS c
INNER JOIN RIDE
ON c.RIDE_ID = RIDE.RIDE_ID
ORDER BY TOTAL_VISIT DESC
LIMIT 3
```

- Output:

	TOTAL_VISIT	RIDE_ID	TYPE_OF_RIDE	DESCRIPTION
▶	77	R005	Thrill	Race down a trio of high-speed loops, then boo...
	69	R010	Family	An into dark steel roller coaster for the thrill see...
	63	R008	Family	Spread your wings and take to the sky as these...

6. **Find old customers'(older than 50 years old) favorite type of ride, see if it is not 'thrill'. Because it is presumed that elder people is not suitable for those rides, we could check to see if it is correct, if not, then the amusement park should also increase their protection to older customers when they take the thrill rides.**

- Code:

```
select count(RIDE_TICKET_ID) as NumberOfLikesFromOld, TYPE_OF_RIDE
FROM(
select b.RIDE_TICKET_ID, b.TYPE_OF_RIDE, YEAR(CURDATE()) - YEAR(DOB)
AS age  from (
select customer.DOB, a.RIDE_TICKET_ID, a.TYPE_OF_RIDE from ticket
inner join (
select ticket_ride.TICKET_ID, ticket_ride.RIDE_TICKET_ID, ride.TYPE_OF_RIDE
from ride
inner join ticket_ride
on ride.RIDE_ID = ticket_ride.RIDE_ID
) as a
on ticket.TICKET_ID = a.TICKET_ID
inner join customer on ticket.C_ID = customer.C_ID
)as b
where YEAR(CURDATE()) - YEAR(DOB) > 50
) as C
group by TYPE_OF_RIDE
ORDER BY NumberOfLikesFromOld DESC
limit 1;
```

- Output:

	NumberOfLikesFromOld	TYPE_OF_RIDE
▶	80	Family

7. **Find the most popular payment method for customers of age older than 50 years old, see if it is that most of them are by cash and offline, because it is presumed that elder people tend to use more traditional way to pay, we can test to see if it is true.**

- Code:

```
select count(PAYMENT_ID) as NumberOfPayments, PAYMENT_METHOD_DESC
FROM (
select b.PAYMENT_ID, b.PAYMENT_METHOD_DESC, b.PURCHASE_MODE,
YEAR(CURDATE()) - YEAR(DOB) AS age
from (
select a.PAYMENT_ID,a.PAYMENT_METHOD_DESC, a.PURCHASE_MODE,
customer.DOB from ticket
inner join (
select payment.PAYMENT_ID, payment.PAYMENT_METHOD_ID,
payment_method.PAYMENT_METHOD_DESC,payment.PURCHASE_MODE from
payment_method
inner join payment on payment_method.PAYMENT_METHOD_ID =
payment.PAYMENT_METHOD_ID
)as a
on ticket.PAYMENT_ID = a.PAYMENT_ID
inner join customer on ticket.C_ID = customer.C_ID
) as b
where  YEAR(CURDATE()) - YEAR(DOB) > 50
) AS c
group by PAYMENT_METHOD_DESC
ORDER BY NumberOfPayments DESC;

select count(PAYMENT_ID) as NumberOfPayments, PURCHASE_MODE FROM (
select b.PAYMENT_ID, b.PAYMENT_METHOD_DESC, b.PURCHASE_MODE,
YEAR(CURDATE()) - YEAR(DOB) AS age
from (
select a.PAYMENT_ID,a.PAYMENT_METHOD_DESC, a.PURCHASE_MODE,
customer.DOB from ticket
inner join (
select payment.PAYMENT_ID, payment.PAYMENT_METHOD_ID,
payment_method.PAYMENT_METHOD_DESC,payment.PURCHASE_MODE from
payment_method
```

```
inner join payment on payment_method.PAYMENT_METHOD_ID =
payment.PAYMENT_METHOD_ID
)as a
on ticket.PAYMENT_ID = a.PAYMENT_ID
inner join customer on ticket.C_ID = customer.C_ID
) as b
where  YEAR(CURDATE()) - YEAR(DOB) > 50
) AS c
group by PURCHASE_MODE
ORDER BY NumberOfPayments desc;
```

- Output:

NumberOfPayments	PAYMENT_METHOD_DESC
35	Debit Card
34	Credit Card
25	Online Payment
19	Cash

NumberOfPayments	PURCHASE_MODE
58	Offline
55	Online

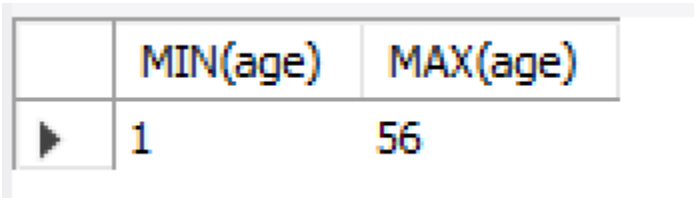
8. **Find the interval of peoples' ages for those who use online Payment to purchase ticket, the amusement park could use this information to detect which customers tend to use internet more, and send them more adds when they are surfing internet.**

- Code:

```
select MIN(age), MAX(age) from (
select YEAR(CURDATE()) - YEAR(DOB) AS age  , PURCHASE_MODE from (
select payment.PURCHASE_MODE, customer.DOB from ticket
inner join payment on ticket.PAYMENT_ID = payment.PAYMENT_ID
inner join customer on ticket.C_ID = customer.C_ID
WHERE PURCHASE_MODE = 'Online'
```

) as a
) as b

- Output:



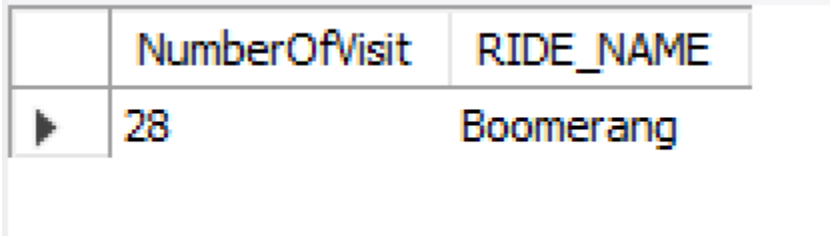
	MIN(age)	MAX(age)
▶	1	56

9. **Find the name of the ride that is most favored in 2019 Nov. The amusement Park could give it** "Best Of the Year Award"

- Code:

```
select count(RIDE_TICKET_ID) AS NumberOfVisit, RIDE_NAME from (
select ride.RIDE_NAME, ticket_ride.RIDE_TICKET_ID, ticket_ride.TIMESTAMP
from ride
inner join ticket_ride on ride.RIDE_ID = ticket_ride.RIDE_ID
where year(TIMESTAMP) = '2019'
) as a
group by RIDE_NAME
ORDER BY NumberOfVisit DESC
limit 1
```

- Output:



	NumberOfVisit	RIDE_NAME
▶	28	Boomerang

10. **Find the amount of visit to dinning facilities from 11:00AM to 2:00AM in 2019, as well as from 3:00 PM - 5:00PM of the same year (see if the first one is indeed greater than second one)**

- Code:

```
select count(a.FACILITY_TICKET_ID) as NumberOFVisits11To2 from
(
select facility.FACILITY_TYPE, ticket_facility.FACILITY_TICKET_ID,
ticket_facility.TIMESTAMP from ticket_facility
inner join facility on ticket_facility.FACILITY_ID = facility.FACILITY_ID
where facility.FACILITY_TYPE = 'Dinning'
)as a
where TIME(TIMESTAMP) >= "11:00:00"
```

```
AND TIME(TIMESTAMP) < "14:00:00"
AND YEAR(TIMESTAMP) = 2019;

select count(a.FACILITY_TICKET_ID) as NumberOFVisits2To5 from
(
select facility.FACILITY_TYPE, ticket_facility.FACILITY_TICKET_ID,
ticket_facility.TIMESTAMP from ticket_facility
inner join facility on ticket_facility.FACILITY_ID = facility.FACILITY_ID
where facility.FACILITY_TYPE = 'Dinning'
)as a
where TIME(TIMESTAMP) >= "14:00:00"
AND TIME(TIMESTAMP) < "17:00:00"
AND YEAR(TIMESTAMP) = 2019;
```

- Output:
  As shown from the output, there might not be correlation between people's time of going to restaurant with the probable hungry time in this amusement park, so restaurant/dinning facilities does not need to pay extra much more attention to their service when the probable hungry time comes, but should also not lose attention even though it is not the perfect time for meal.

	NumberOFVisits11To2
▶	86

	NumberOFVisits2To5
▶	96