DMDD Final Take Home Exam
Zixiao Wang 20200412

DMDD Final Take Home Exam

Summary for this course

Zixiao:

For me, this is kind of an unforgettable experience.

In a mental view, I grow a lot. At first, I was just thinking about learning some basic SQL knowledge. From the beginning of this course, I was shocked by the 'innovative final project' requirement. At first, I thought that this course will only cover the database design and SQL knowledge. So it will be very difficult in order to finish this final project. Because some 'innovative ideas' sometimes combine with some cutting-edge technologies. So I have to learn a lot of knowledge by myself.

Gradually, I found that it is not the technology that is important, it is the idea. Some very old technology can also help you to implement your idea. Technology is just a kind of tool. So I gradually stopped focusing on the technology itself but focused on the idea. When I think this idea is worth paying for, I just go back and consider which technologies to use. So 'How to think' is the first lesson I learned from this class.

Then, after me and my teammate decided on the topic. I found I was facing another problem: How to divide the work. At first, nobody knew how to implement our topic. It was like there were a large number of possible methods that we can use in our project.

Starting from here, I found something called 'roadmap' from the class. I learned how to use this tool to divide the project and how to regulate the workflow for the project. This is because a roadmap can help everyone understand every single piece of the project, and during this period, the roadmap can also help us standardize the work data flow of each piece. This is like black-boxing the module. After that, we assigned work, and everyone could focus on the part that he was responsible for. Because the data input and output are standardized, in the end, we only need to stitch together everyone's achievements. So 'RoadMap' is the second thing I learned from this class.

Then during the implementation, I learned how to use ERD, how to use MySQL. And because I was in charge of algorithms. I learned a lot of theories like BFS, Dijkstra's algorithm, A* algorithm, WiFi-based indoor positioning tech, and beacon-based indoor positioning tech. And I used Python a lot before, so I got a chance to improve my Python skill during this time. I learned how to use the database connection, how to code with MySQL functions, store procedures, triggers, views. You know, writing more than one thousand rows of code really gives me a better understanding of Python. So these are the third things I learned from this class.

Apart from these three things, I think the most important thing I learned is courage, the courage that gives me confidence and passion to overcome any kind of difficulties.

Thanks for the friendship with my teammates, this was really an impressive experience.

```
import pandas as pd
%load_ext sql
%sql mysql+pymysql://root:fjwwzx970814@localhost/mydb
```

```
'Connected: root@mydb'
```

```
import sqlalchemy as sqlManager
import random
from datetime import date
```

```
# Create connection with database
connection = sqlManager.create_engine('mysql+pymysql://root:fjwwzx970814@localhost/mydb')
```

Abstract

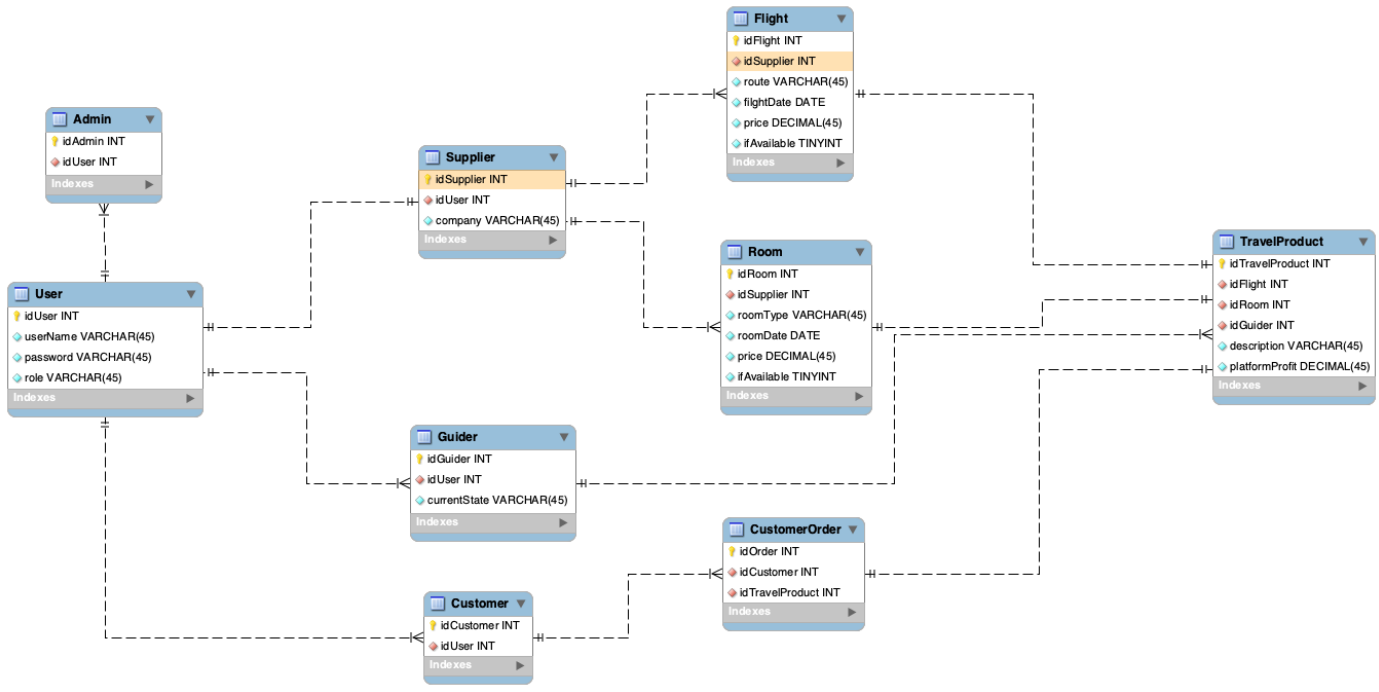
1. Create 4-6 Tables each with 25-50 records for the following demonstration. (You may use website that generate random data sample as needed)
2. Create a simple ER diagram showing mapping and relationship of these tables.
3. For each of the following topics, create a SQL-script (in MySQL or MS SQL) and a power point slide (1- 5 slides for each of the set below) explaining what the code does and its outcome. (Hint: Imagine you are a TA for INFO6210 and you are conducting a code demo for a new set of INFO6210 students, these would be what you provide to the students and demonstrate during class)
    1. Create DB, creates tables, create sample records/values/tuples, USE, and Drop DB
    2. Use DB, create tables with table options (NULL, Key, etc.), data types, insert, index and set
    3. Select – From – Where – Group By – Having - Order By – Limit
    4. Select 10 functions in SQL statement of your choice (including but not limited to Comparison, control flow, cast, string, numeric, date/time manipulation, calculation, conversion, aggregate functions, etc.)
    5. Perform JOIN operation including INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, CROSS JOIN operations and display the sample result using SELECT statement
    6. Perform Queries and Sub-queries operations and display the result using SELECT statement
    7. Create four different users (Admin, Tester, Developer and User) with different privilege settings and display access differentiation among four users. Use GRANT and REVOKE command to give/remove additional access/rights to DB.

8. Create two examples of STORED PROCEDURE and display the results.
9. Perform COMMIT and ROLLBACK operations with SAVEPOINT/ROLLBACK TO SAVEPOINT. Include the use of LOCK/UNLOCK Table operations. Display result of each operation.
10. Perform Triggers operations using INSERT, UPDATE, DELETE and display result
11. Perform three examples of VIEW operation (1) Simple, 2) and 3) with increasing Complexities) Create a backup plan of your choice, state your assumptions/constraints and explain your reasoning. Print your selection on a sample monthly calendar (say 30 day) with FB for Full Backup and IB for Incremental Backup

4. Create 10 True or False questions with solution from any topic covered in this class
5. Create 10 multiple choice questions including solution covering queries, subqueries, trigger, store procedures, math functions, backup, privilege and any other topics covered in the class.
6. Create 3 questions including solution and associated diagrams that cover normalization from 1NF, 2NF, 3NF and BCNF
7. Create 2 questions including solution and associated diagrams that cover ERD and EERD concepts
8. Create 5 questions including solution and code covering the SQL coding skills and concept
9. Write a team evaluation (for each of your team members—2-3 sentences for each team member) including who do you think is the most valuable team member and why; Who did what according to your own observations? Rank everyone on your team including yourself (1 is the best, 2 is 2nd best, etc.) and why they are ranked in that order (no equal contribution will be an acceptable answer); what works best for your team and what did not? Rank all the team in class including your team.
10. If you are given an opportunity to be a paid TA or non-paid/volunteer in Fall 2020/Spring 2021, would you be interested (Do include if you are interested in volunteer vs paid position?) If you are interested, what skillsets and quality would make you a successful TA or a part of the team?

Content

Table Creation and Records Insertion



Basicly, I create 9 tables in order to complete travel agent system

Table Creation

```
-- MySQL Script generated by MySQL Workbench
-- Mon Apr 13 12:23:21 2020
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
```

```
-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`User`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`User` (
  `idUser` INT NOT NULL AUTO_INCREMENT,
  `userName` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `role` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idUser`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Supplier`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Supplier` (
  `idSupplier` INT NOT NULL AUTO_INCREMENT,
  `idUser` INT NOT NULL,
  `company` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idSupplier`),
  INDEX `idUserForSupplier_idx` (`idUser` ASC),
  CONSTRAINT `idUserForSupplier`
    FOREIGN KEY (`idUser`)
    REFERENCES `mydb`.`User` (`idUser`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Flight`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Flight` (
  `idFlight` INT NOT NULL AUTO_INCREMENT,
  `idSupplier` INT NOT NULL,
  `route` VARCHAR(45) NOT NULL,
  `filghtDate` DATE NOT NULL,
  `price` DECIMAL(45) NOT NULL,
  `ifAvailable` TINYINT NOT NULL DEFAULT 1,
  PRIMARY KEY (`idFlight`),
  INDEX `idSupplierForFlights_idx` (`idSupplier` ASC),
  CONSTRAINT `idSupplierForFlights`
    FOREIGN KEY (`idSupplier`)
    REFERENCES `mydb`.`Supplier` (`idSupplier`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Room`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Room` (
  `idRoom` INT NOT NULL AUTO_INCREMENT,
  `idSupplier` INT NOT NULL,
  `roomType` VARCHAR(45) NOT NULL,
  `roomDate` DATE NOT NULL,
  `price` DECIMAL(45) NOT NULL,
  `ifAvailable` TINYINT NOT NULL DEFAULT 1,
  PRIMARY KEY (`idRoom`),
  INDEX `idSupplierForRoom_idx` (`idSupplier` ASC),
  CONSTRAINT `idSupplierForRoom`
    FOREIGN KEY (`idSupplier`)
    REFERENCES `mydb`.`Supplier` (`idSupplier`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
```

```
-- Table `mydb`.`Admin`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Admin` (
  `idAdmin` INT NOT NULL AUTO_INCREMENT,
  `idUser` INT NOT NULL,
  PRIMARY KEY (`idAdmin`),
  INDEX `idUserForAdmin_idx` (`idUser` ASC),
  CONSTRAINT `idUserForAdmin`
    FOREIGN KEY (`idUser`)
    REFERENCES `mydb`.`User` (`idUser`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Guider`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Guider` (
  `idGuider` INT NOT NULL AUTO_INCREMENT,
  `idUser` INT NOT NULL,
  `currentState` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idGuider`),
  INDEX `idUserForGuider_idx` (`idUser` ASC),
  CONSTRAINT `idUserForGuider`
    FOREIGN KEY (`idUser`)
    REFERENCES `mydb`.`User` (`idUser`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Customer` (
  `idCustomer` INT NOT NULL AUTO_INCREMENT,
  `idUser` INT NOT NULL,
  PRIMARY KEY (`idCustomer`),
  INDEX `idUserForCustomer_idx` (`idUser` ASC),
  CONSTRAINT `idUserForCustomer`
    FOREIGN KEY (`idUser`)
    REFERENCES `mydb`.`User` (`idUser`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`TravelProduct`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`TravelProduct` (
  `idTravelProduct` INT NOT NULL AUTO_INCREMENT,
  `idFlight` INT NOT NULL,
  `idRoom` INT NOT NULL,
  `idGuider` INT NOT NULL,
  `description` VARCHAR(45) NOT NULL,
  `platformProfit` DECIMAL(45) NOT NULL,
  PRIMARY KEY (`idTravelProduct`),
  INDEX `idFlightForTravelProduct_idx` (`idFlight` ASC),
  INDEX `idRoomForTravelProduct_idx` (`idRoom` ASC),
  INDEX `idGuiderForTravelProduct_idx` (`idGuider` ASC),
  CONSTRAINT `idFlightForTravelProduct`
    FOREIGN KEY (`idFlight`)
    REFERENCES `mydb`.`Flight` (`idFlight`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `idRoomForTravelProduct`
    FOREIGN KEY (`idRoom`)
    REFERENCES `mydb`.`Room` (`idRoom`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `idGuiderForTravelProduct`
    FOREIGN KEY (`idGuider`)
    REFERENCES `mydb`.`Guider` (`idGuider`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
```

```
-- Table `mydb`.`Order`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Order` (
  `idOrder` INT NOT NULL AUTO_INCREMENT,
  `idCustomer` INT NOT NULL,
  `idTravelProduct` INT NOT NULL,
  PRIMARY KEY (`idOrder`),
  INDEX `idCustomerForOrder_idx` (`idCustomer` ASC),
  INDEX `idTravelProductFroOrder_idx` (`idTravelProduct` ASC),
  CONSTRAINT `idCustomerForOrder`
    FOREIGN KEY (`idCustomer`)
    REFERENCES `mydb`.`Customer` (`idCustomer`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `idTravelProductFroOrder`
    FOREIGN KEY (`idTravelProduct`)
    REFERENCES `mydb`.`TravelProduct` (`idTravelProduct`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


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

Records Insertion

Records Insertion Results

```
%%sql
select * from User;
```

```
 * mysql+pymysql://root:***@localhost/mydb
25 rows affected.
```

idUser	userName	password	role
1	user_1	password_1	Customer
2	user_2	password_2	Customer
3	user_3	password_3	Customer
4	user_4	password_4	Customer
5	user_5	password_5	Customer
6	user_6	password_6	Customer
7	user_7	password_7	Customer
8	user_8	password_8	Customer
9	user_9	password_9	Customer
10	user_10	password_10	Customer
11	user_11	password_11	Admin
12	user_12	password_12	Admin
13	user_13	password_13	Admin
14	user_14	password_14	Admin
15	user_15	password_15	Admin
16	user_16	password_16	Supplier
17	user_17	password_17	Supplier
18	user_18	password_18	Supplier
19	user_19	password_19	Supplier
20	user_20	password_20	supplier
21	user_21	password_21	Guider

22	user_22	password_22	Guider
23	user_23	password_23	Guider
24	user_24	password_24	Guider
25	user_25	password_25	Guider

```
%%sql
select * from Admin
```

```
 * mysql+pymysql://root:***@localhost/mydb
5 rows affected.
```

idAdmin	idUser
11	11
12	12
13	13
14	14
15	15

```
%%sql
select * from Customer
```

```
 * mysql+pymysql://root:***@localhost/mydb
10 rows affected.
```

idCustomer	idUser
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

```
%%sql
select * from Supplier
```

```
 * mysql+pymysql://root:***@localhost/mydb
5 rows affected.
```

idSupplier	idUser	company
16	16	company_16
17	17	company_17
18	18	company_18
19	19	company_19
20	20	company_20

```
%%sql
select * from Guider
```

```
 * mysql+pymysql://root:***@localhost/mydb
5 rows affected.
```

idGuider	idUser	currentState
21	21	Not working
22	22	Not working
23	23	Not working
24	24	Not working
25	25	Not working

```
%%sql
select * from Flight
```

```
 * mysql+pymysql://root:***@localhost/mydb
25 rows affected.
```

idFlight	idSupplier	route	flightDate	price	ifAvailable
1	16	China-America	2020-06-17	16044	1
2	17	China-America	2020-06-10	11384	1
3	18	China-Japen	2020-05-25	12000	1
4	19	China-Japen	2020-04-19	17422	1
5	20	China-America	2020-06-19	15136	1
6	16	China-America	2020-04-10	10575	1
7	17	China-Japen	2020-05-21	16731	1
8	18	China-Japen	2020-05-13	15579	1
9	19	China-Japen	2020-06-01	19223	1
10	20	China-France	2020-06-10	14325	1
11	16	China-France	2020-04-15	10000	1
12	17	China-France	2020-05-24	12045	1
13	18	China-America	2020-04-11	13643	1
14	19	China-America	2020-04-13	11813	1
15	20	China-Japen	2020-06-27	14727	1
16	16	China-America	2020-04-05	14129	1
17	17	China-France	2020-04-30	13298	1
18	18	China-America	2020-06-17	13064	1
19	19	China-France	2020-05-27	10642	1
20	20	China-Japen	2020-04-23	14763	1
21	16	China-Japen	2020-05-06	18436	1
22	17	China-France	2020-04-28	11112	1
23	18	China-Japen	2020-06-22	17515	1
24	19	China-France	2020-04-14	13450	1
25	20	China-France	2020-04-25	11585	1

```
%%sql
select * from Room
```

```
 * mysql+pymysql://root:***@localhost/mydb
21 rows affected.
```

idRoom	idSupplier	roomType	roomDate	price	ifAvailable
7	16	single	2020-05-09	341	1
13	16	double	2020-06-13	281	1
14	17	single	2020-05-09	241	1
15	18	double	2020-05-05	485	1
16	19	single	2020-05-05	382	1
17	20	double	2020-04-16	131	1
18	16	single	2020-04-24	402	1
19	17	single	2020-06-21	144	1
20	18	single	2020-05-25	345	1
21	19	single	2020-05-17	487	1
22	20	single	2020-06-26	151	1
28	16	double	2020-04-29	459	1
29	17	single	2020-05-13	109	1
30	18	double	2020-05-11	288	1
31	19	single	2020-04-03	409	1
32	20	single	2020-05-23	307	1
38	16	single	2020-06-18	338	1
39	17	double	2020-06-25	215	1
40	18	double	2020-04-15	135	1
41	19	single	2020-06-05	274	1
42	20	double	2020-06-06	307	1

```
%%sql
select * from CustomerOrder
```

```
 * mysql+pymysql://root:***@localhost/mydb
50 rows affected.
```

idOrder	idCustomer	idTravelProduct
1	6	18
2	9	14
3	4	18
4	1	4
5	7	18
6	8	14
7	6	3
8	9	15
9	6	17
10	5	17
11	10	14

12	2	14
13	8	18
14	9	5
15	3	19
16	5	9
17	7	16
18	4	2
19	5	3
20	1	3
21	2	16
22	2	1
23	5	19
24	7	9
25	10	1
26	6	8
27	4	20
28	9	19
29	5	13
30	1	18
31	1	2
32	1	7
33	5	8
34	8	9
35	10	4
36	1	7
37	4	13
38	7	12
39	8	1
40	2	8
41	3	10
42	1	11
43	5	11
44	7	14
45	2	11
46	1	19
47	10	8
48	2	20
49	4	18
50	3	2

```
%%sql
select * from TravelProduct
```

```
 * mysql+pymysql://root:***@localhost/mydb
20 rows affected.
```

idTravelProduct	idFlight	idRoom	idGuider	description	platformProfit

1	19	30	25	Good	628
2	12	22	21	Mid	449
3	20	21	25	Good	658
4	1	7	25	Mid	761
5	6	15	25	Good	520
6	17	29	25	Perfect	437
7	14	28	25	Mid	633
8	22	14	23	Mid	416
9	23	15	25	Good	518
10	5	13	25	Perfect	483
11	20	40	23	Perfect	338
12	23	19	23	Good	431
13	1	7	22	Mid	432
14	2	15	23	Perfect	676
15	23	21	21	Good	452
16	19	19	24	Mid	552
17	9	7	22	Good	717
18	5	15	25	Mid	307
19	18	30	23	Mid	662
20	2	41	25	Mid	425

```
%%sql
show tables
```

```
 * mysql+pymysql://root:***@localhost/mydb
11 rows affected.
```

Tables_in_mydb
Admin
Customer
CustomizedTravelPlan
Flight
Guider
Order
Room
Supplier
Suuplier
TravelProduct
User

The code I used for simulation and insertion

User insertion

Customers

```
user_data = []
for i in range(10):
    user_data.append(
        ['user_'+str(i+1),'password_'+str(i+1),'Customer']
    )
user_columns = ['userName','password','role']
```

```
user_df = pd.DataFrame(data = user_data,columns = user_columns)
```

```
# Insert data into database
pd.io.sql.to_sql(user_df,'User',connection,schema='mydb',if_exists='append',index=False)
```

```
%%sql
select * from User
```

```
 * mysql+pymysql://root:***@localhost/mydb
10 rows affected.
```

idUser	userName	password	role
1	user_1	password_1	Customer
2	user_2	password_2	Customer
3	user_3	password_3	Customer
4	user_4	password_4	Customer
5	user_5	password_5	Customer
6	user_6	password_6	Customer
7	user_7	password_7	Customer
8	user_8	password_8	Customer
9	user_9	password_9	Customer
10	user_10	password_10	Customer

```
customer_data = []
for i in range(10):
    customer_data.append(
        [i+1,i+1]
    )
customer_columns = ['idCustomer','idUser']
```

```
customer_df = pd.DataFrame(data = customer_data,columns = customer_columns)
```

```
# Insert data into database
pd.io.sql.to_sql(customer_df,'Customer',connection,schema='mydb',if_exists='append',index=False)
```

```
%%sql
select * from Customer
```

```
 * mysql+pymysql://root:***@localhost/mydb
10 rows affected.
```

idCustomer	idUser
1	1
2	2
3	3
4	4
5	5

6	6
7	7
8	8
9	9
10	10

Admin

```
admin_data = []
for i in range(10,15):
    admin_data.append(
        ['user_'+str(i+1),'password_'+str(i+1),'Admin']
    )
admin_columns = ['userName','password','role']
```

```
admin_df = pd.DataFrame(data = admin_data,columns = admin_columns)
```

```
# Insert data into database
pd.io.sql.to_sql(admin_df,'User',connection,schema='mydb',if_exists='append',index=False)
```

```
%%sql
select * from User
```

```
 * mysql+pymysql://root:***@localhost/mydb
15 rows affected.
```

idUser	userName	password	role
1	user_1	password_1	Customer
2	user_2	password_2	Customer
3	user_3	password_3	Customer
4	user_4	password_4	Customer
5	user_5	password_5	Customer
6	user_6	password_6	Customer
7	user_7	password_7	Customer
8	user_8	password_8	Customer
9	user_9	password_9	Customer
10	user_10	password_10	Customer
11	user_11	password_11	Admin
12	user_12	password_12	Admin
13	user_13	password_13	Admin
14	user_14	password_14	Admin
15	user_15	password_15	Admin

```
admin_data = []
for i in range(10,15):
    admin_data.append(
        [i+1,i+1]
    )
admin_columns = ['idAdmin','idUser']
```

```
admin_df = pd.DataFrame(data = admin_data,columns = admin_columns)
```

```
# Insert data into database
pd.io.sql.to_sql(admin_df,'admin',connection,schema='mydb',if_exists='append',index=False)
```

```
%%sql
select * from admin
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
5 rows affected.
```

idAdmin	idUser
11	11
12	12
13	13
14	14
15	15

Supplier

```
supplier_data = []
for i in range(15,20):
    supplier_data.append(
        ['user_'+str(i+1),'password_'+str(i+1),'supplier']
    )
supplier_columns = ['userName','password','role']
```

```
supplier_df = pd.DataFrame(data = supplier_data,columns = supplier_columns)
```

```
# Insert data into database
pd.io.sql.to_sql(supplier_df,'User',connection,schema='mydb',if_exists='append',index=False)
```

```
%%sql
select * from User
```

```
 * mysql+pymysql://root:***@localhost/mydb
20 rows affected.
```

idUser	userName	password	role
1	user_1	password_1	Customer
2	user_2	password_2	Customer
3	user_3	password_3	Customer
4	user_4	password_4	Customer
5	user_5	password_5	Customer
6	user_6	password_6	Customer
7	user_7	password_7	Customer
8	user_8	password_8	Customer
9	user_9	password_9	Customer
10	user_10	password_10	Customer

11	user_11	password_11	Admin
12	user_12	password_12	Admin
13	user_13	password_13	Admin
14	user_14	password_14	Admin
15	user_15	password_15	Admin
16	user_16	password_16	supplier
17	user_17	password_17	supplier
18	user_18	password_18	supplier
19	user_19	password_19	supplier
20	user_20	password_20	supplier

```
supplier_data = []
for i in range(15,20):
    supplier_data.append(
        [i+1,i+1,'company_'+str(i+1)]
    )
supplier_columns = ['idSupplier','idUser','company']
```

```
supplier_df = pd.DataFrame(data = supplier_data,columns = supplier_columns)
```

```
# Insert data into database
pd.io.sql.to_sql(supplier_df,'supplier',connection,schema='mydb',if_exists='append',index=False)
```

```
%%sql
select * from Supplier
```

```
 * mysql+pymysql://root:***@localhost/mydb
5 rows affected.
```

idSupplier	idUser	company
16	16	company_16
17	17	company_17
18	18	company_18
19	19	company_19
20	20	company_20

Guider

```
guider_data = []
for i in range(20,25):
    guider_data.append(
        ['user_'+str(i+1),'password_'+str(i+1),'Guider']
    )
guider_columns = ['userName','password','role']
```

```
guider_df = pd.DataFrame(data = guider_data,columns = guider_columns)
```

```
# Insert data into database
pd.io.sql.to_sql(guider_df,'User',connection,schema='mydb',if_exists='append',index=False)
```

```
%%sql
select * from User
```

```
 * mysql+pymysql://root:***@localhost/mydb
25 rows affected.
```

idUser	userName	password	role
1	user_1	password_1	Customer
2	user_2	password_2	Customer
3	user_3	password_3	Customer
4	user_4	password_4	Customer
5	user_5	password_5	Customer
6	user_6	password_6	Customer
7	user_7	password_7	Customer
8	user_8	password_8	Customer
9	user_9	password_9	Customer
10	user_10	password_10	Customer
11	user_11	password_11	Admin
12	user_12	password_12	Admin
13	user_13	password_13	Admin
14	user_14	password_14	Admin
15	user_15	password_15	Admin
16	user_16	password_16	supplier
17	user_17	password_17	supplier
18	user_18	password_18	supplier
19	user_19	password_19	supplier
20	user_20	password_20	supplier
21	user_21	password_21	Guider
22	user_22	password_22	Guider
23	user_23	password_23	Guider
24	user_24	password_24	Guider
25	user_25	password_25	Guider

```
guider_data = []
for i in range(20,25):
    guider_data.append(
        [i+1,i+1,'Not working']
    )
guider_columns = ['idGuider','idUser','currentState']
```

```
guider_df = pd.DataFrame(data = guider_data,columns = guider_columns)
```

```
# Insert data into database
pd.io.sql.to_sql(guider_df,'Guider',connection,schema='mydb',if_exists='append',index=False)
```

```
%%sql
select * from Guider
```

```
 * mysql+pymysql://root:***@localhost/mydb
5 rows affected.
```

idGuider	idUser	currentState
21	21	Not working
22	22	Not working
23	23	Not working
24	24	Not working
25	25	Not working

Room & Flight Insertion

room

```
%%sql
select * from Supplier
```

```
 * mysql+pymysql://root:***@localhost/mydb
5 rows affected.
```

idSupplier	idUser	company
16	16	company_16
17	17	company_17
18	18	company_18
19	19	company_19
20	20	company_20

```
sql = 'select idSupplier from Supplier'
idSupplier = pd.read_sql(sql,connection)
```

```
month = [i for i in range(4,7)]
day = [i for i in range(1,31)]
```

```
for supplier in idSupplier.idSupplier:
    roomType = random.choice(['single','double'])
    random_date = date(2020,random.choice(month),random.choice(day))
    price = random.randint(100,500)
    sql = 'call insert_room('+str(supplier)+',"'+str(roomType)+'","'+str(random_date)+'",'+str(price)+');'
    print(sql)
```

```
call insert_room(16,"single","2020-06-18",338);
call insert_room(17,"double","2020-06-25",215);
call insert_room(18,"double","2020-04-15",135);
call insert_room(19,"single","2020-06-05",274);
call insert_room(20,"double","2020-06-06",307);
```

Flight

```
%%sql
select * from Supplier
```

```
 * mysql+pymysql://root:***@localhost/mydb
5 rows affected.
```

idSupplier	idUser	company
16	16	company_16
17	17	company_17
18	18	company_18
19	19	company_19
20	20	company_20

```
month = [i for i in range(4,7)]
day = [i for i in range(1,31)]
```

```
for supplier in idSupplier.idSupplier:
    route = random.choice(['China-America','China-Japen','China-France'])
    random_date = date(2020,random.choice(month),random.choice(day))
    price = random.randint(10000,20000)
    sql = 'call insert_flight('+str(supplier)+',"'+str(route)+'","'+str(random_date)+'",'+str(price)+');'
    print(sql)
```

```
call insert_flight(16,"China-Japen","2020-05-06",18436);
call insert_flight(17,"China-France","2020-04-28",11112);
call insert_flight(18,"China-Japen","2020-06-22",17515);
call insert_flight(19,"China-France","2020-04-14",13450);
call insert_flight(20,"China-France","2020-04-25",11585);
```

Travel product Insertion

```
%%sql
select * from room
```

```
 * mysql+pymysql://root:***@localhost/mydb
21 rows affected.
```

idRoom	idSupplier	roomType	roomDate	price	ifAvailable
7	16	single	2020-05-09	341	1
13	16	double	2020-06-13	281	1
14	17	single	2020-05-09	241	1
15	18	double	2020-05-05	485	1
16	19	single	2020-05-05	382	1
17	20	double	2020-04-16	131	1
18	16	single	2020-04-24	402	1
19	17	single	2020-06-21	144	1
20	18	single	2020-05-25	345	1
21	19	single	2020-05-17	487	1
22	20	single	2020-06-26	151	1
28	16	double	2020-04-29	459	1
29	17	single	2020-05-13	109	1
30	18	double	2020-05-11	288	1
31	19	single	2020-04-03	409	1

32	20	single	2020-05-23	307	1
38	16	single	2020-06-18	338	1
39	17	double	2020-06-25	215	1
40	18	double	2020-04-15	135	1
41	19	single	2020-06-05	274	1
42	20	double	2020-06-06	307	1

```
%%sql
select * from flight
```

```
 * mysql+pymysql://root:***@localhost/mydb
25 rows affected.
```

idFlight	idSupplier	route	flightDate	price	ifAvailable
1	16	China-America	2020-06-17	16044	1
2	17	China-America	2020-06-10	11384	1
3	18	China-Japen	2020-05-25	12000	1
4	19	China-Japen	2020-04-19	17422	1
5	20	China-America	2020-06-19	15136	1
6	16	China-America	2020-04-10	10575	1
7	17	China-Japen	2020-05-21	16731	1
8	18	China-Japen	2020-05-13	15579	1
9	19	China-Japen	2020-06-01	19223	1
10	20	China-France	2020-06-10	14325	1
11	16	China-France	2020-04-15	10000	1
12	17	China-France	2020-05-24	12045	1
13	18	China-America	2020-04-11	13643	1
14	19	China-America	2020-04-13	11813	1
15	20	China-Japen	2020-06-27	14727	1
16	16	China-America	2020-04-05	14129	1
17	17	China-France	2020-04-30	13298	1
18	18	China-America	2020-06-17	13064	1
19	19	China-France	2020-05-27	10642	1
20	20	China-Japen	2020-04-23	14763	1
21	16	China-Japen	2020-05-06	18436	1
22	17	China-France	2020-04-28	11112	1
23	18	China-Japen	2020-06-22	17515	1
24	19	China-France	2020-04-14	13450	1
25	20	China-France	2020-04-25	11585	1

```
%%sql
select * from Guider
```

```
 * mysql+pymysql://root:***@localhost/mydb
5 rows affected.
```

idGuider	idUser	currentState
21	21	Not working

22	22	Not working
23	23	Not working
24	24	Not working
25	25	Not working

```
sql = 'select idFlight from Flight'
flights = pd.read_sql(sql,connection).idFlight
```

```
sql = 'select idRoom from Room'
rooms = pd.read_sql(sql,connection).idRoom
```

```
sql = 'select idGuider from Guider'
guiders = pd.read_sql(sql,connection).idGuider
```

```
for i in range(20):
    room = random.choice(rooms)
    flight = random.choice(flights)
    guider = random.choice(guiders)
    profit = random.randint(300,800)
    description = random.choice(['Good','Mid','Perfect'])

    sql = 'call
insert_travel_product('+str(flight)+','+str(room)+','+str(guider)+",'"+description+"',"+str(profit)+');'
    print(sql)
```

```
call insert_travel_product(19,30,25,"Good",628);
call insert_travel_product(12,22,21,"Mid",449);
call insert_travel_product(20,21,25,"Good",658);
call insert_travel_product(1,7,25,"Mid",761);
call insert_travel_product(6,15,25,"Good",520);
call insert_travel_product(17,29,25,"Perfect",437);
call insert_travel_product(14,28,25,"Mid",633);
call insert_travel_product(22,14,23,"Mid",416);
call insert_travel_product(23,15,25,"Good",518);
call insert_travel_product(5,13,25,"Perfect",483);
call insert_travel_product(20,40,23,"Perfect",338);
call insert_travel_product(23,19,23,"Good",431);
call insert_travel_product(1,7,22,"Mid",432);
call insert_travel_product(2,15,23,"Perfect",676);
call insert_travel_product(23,21,21,"Good",452);
call insert_travel_product(19,19,24,"Mid",552);
call insert_travel_product(9,7,22,"Good",717);
call insert_travel_product(5,15,25,"Mid",307);
call insert_travel_product(18,30,23,"Mid",662);
call insert_travel_product(2,41,25,"Mid",425);
```

Order Insertion

```
%%sql
select * from TravelProduct
```

```
 * mysql+pymysql://root:***@localhost/mydb
20 rows affected.
```

idTravelProduct	idFlight	idRoom	idGuider	description	platformProfit
1	19	30	25	Good	628
2	12	22	21	Mid	449
3	20	21	25	Good	658

4	1	7	25	Mid	761
5	6	15	25	Good	520
6	17	29	25	Perfect	437
7	14	28	25	Mid	633
8	22	14	23	Mid	416
9	23	15	25	Good	518
10	5	13	25	Perfect	483
11	20	40	23	Perfect	338
12	23	19	23	Good	431
13	1	7	22	Mid	432
14	2	15	23	Perfect	676
15	23	21	21	Good	452
16	19	19	24	Mid	552
17	9	7	22	Good	717
18	5	15	25	Mid	307
19	18	30	23	Mid	662
20	2	41	25	Mid	425

```
%%sql
select * from Customer
```

```
 * mysql+pymysql://root:***@localhost/mydb
10 rows affected.
```

idCustomer	idUser
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

```
sql = 'select idCustomer from Customer'
customers = pd.read_sql(sql,connection).idCustomer
```

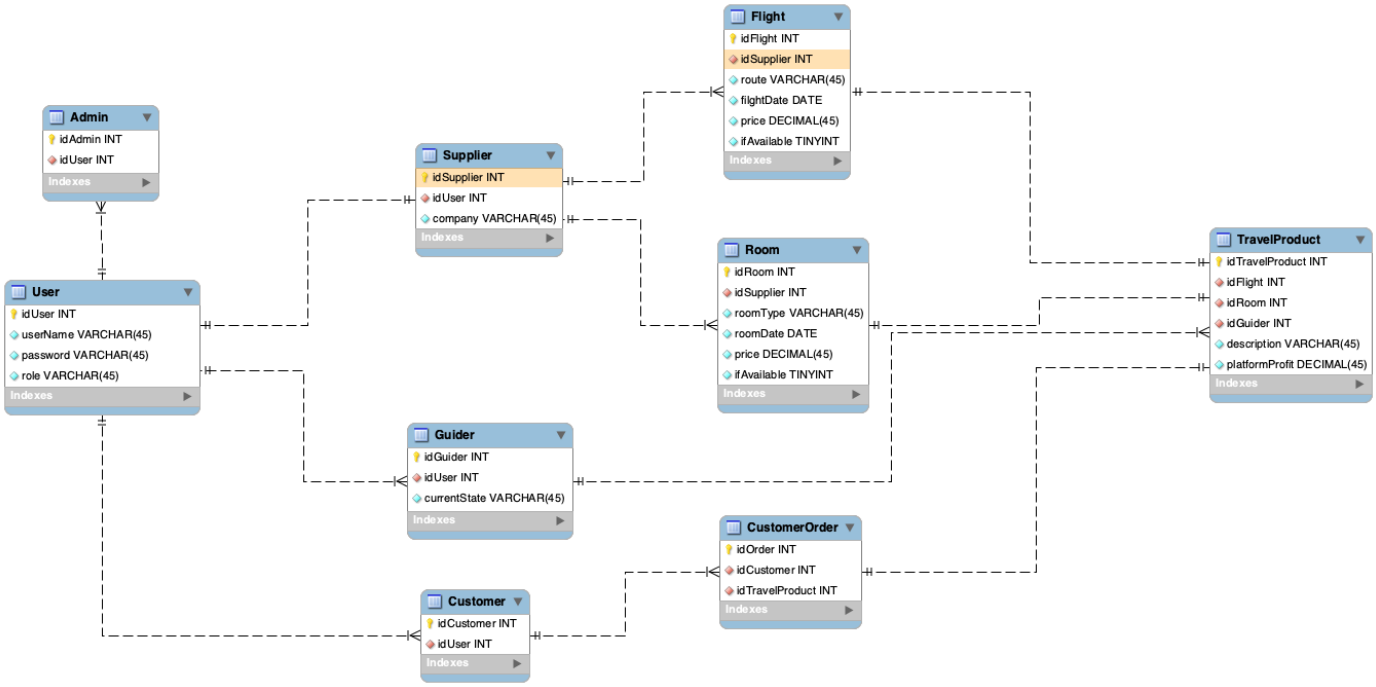
```
sql = 'select idTravelProduct from TravelProduct'
travel_products = pd.read_sql(sql,connection).idTravelProduct
```

```
for i in range(50):
    customer = random.choice(customers)
    travel_product = random.choice(travel_products)

    sql = 'call insert_order('+str(customer)+','+str(travel_product)+');'
    print(sql)
```

```
call insert_order(6,18);
call insert_order(9,14);
call insert_order(4,18);
call insert_order(1,4);
call insert_order(7,18);
call insert_order(8,14);
call insert_order(6,3);
call insert_order(9,15);
call insert_order(6,17);
call insert_order(5,17);
call insert_order(10,14);
call insert_order(2,14);
call insert_order(8,18);
call insert_order(9,5);
call insert_order(3,19);
call insert_order(5,9);
call insert_order(7,16);
call insert_order(4,2);
call insert_order(5,3);
call insert_order(1,3);
call insert_order(2,16);
call insert_order(2,1);
call insert_order(5,19);
call insert_order(7,9);
call insert_order(10,1);
call insert_order(6,8);
call insert_order(4,20);
call insert_order(9,19);
call insert_order(5,13);
call insert_order(1,18);
call insert_order(1,2);
call insert_order(1,7);
call insert_order(5,8);
call insert_order(8,9);
call insert_order(10,4);
call insert_order(1,7);
call insert_order(4,13);
call insert_order(7,12);
call insert_order(8,1);
call insert_order(2,8);
call insert_order(3,10);
call insert_order(1,11);
call insert_order(5,11);
call insert_order(7,14);
call insert_order(2,11);
call insert_order(1,19);
call insert_order(10,8);
call insert_order(2,20);
call insert_order(4,18);
call insert_order(3,2);
```

ERD Diagram



SQL Script for each Topic

1. Create DB, creates tables, create sample records/values/tuples, USE, and Drop DB
2. Use DB, create tables with table options (NULL, Key, etc.), data types, insert, index and set
3. Select – From – Where – Group By – Having - Order By – Limit
4. Select 10 functions in SQL statement of your choice (including but not limited to Comparison, control flow, cast, string, numeric, date/time manipulation, calculation, conversion, aggregate functions, etc.)
5. Perform JOIN operation including INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, CROSS JOIN operations and display the sample result using SELECT statement
6. Perform Queries and Sub-queries operations and display the result using SELECT statement
7. Create four different users (Admin, Tester, Developer and User) with different privilege settings and display access differentiation among four users. Use GRANT and REVOKE command to give/remove additional access/rights to DB.
8. Create two examples of STORED PROCEDURE and display the results.
9. Perform COMMIT and ROLLBACK operations with SAVEPOINT/ROLLBACK TO SAVEPOINT. Include the use of LOCK/UNLOCK Table operations. Display result of each operation.
10. Perform Triggers operations using INSERT, UPDATE, DELETE and display result
11. Perform three examples of VIEW operation (1) Simple, 2) and 3) with increasing Complexities) Create a backup plan of your choice, state your assumptions/constraints and explain your reasoning. Print your selection on a sample monthly calendar (say 30 day) with FB for Full Backup and IB for Incremental Backup

1. Create DB, creates tables, create sample records/values/tuples, USE, and Drop DB

Abstract example:

```
-- Create DB, use DB and drop DB
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;
DROP DATABASE `mydb`;

-- Create tables. Take 'User' table as example
CREATE TABLE IF NOT EXISTS `mydb`.`User` (
  `idUser` INT NOT NULL AUTO_INCREMENT,
  `userName` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `role` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idUser`))

-- Create records in tables. Take 'User' table as example
INSERT INTO `mydb`.`User` (`idUser`, `userName`, `password`, `role`) VALUES ('26', 'user_26', 'password_26',
'Guider');
```

1.1 Create DB

Statement:

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
```

Here is a basic creation for DB

```
%%sql
create schema test;
```

```
 * mysql+pymysql://root:***@localhost/mydb
(pymysql.err.ProgrammingError) (1007, "Can't create database 'test'; database exists")
[SQL: create schema test;]
(Background on this error at: http://sqlalche.me/e/f405)
```

Then we can check if our database has been created

```
%%sql
show databases;
```

```
 * mysql+pymysql://root:***@localhost/mydb
6 rows affected.
```

Database
information_schema
mydb
mysql
performance_schema
sys
test

And if you are not sure if the database is exist

```
%%sql
-- if we create database directly, we will got a error 'Can't create database 'test'; database exists'
create schema `test`
```

```
 * mysql+pymysql://root:***@localhost/mydb
(pymysql.err.ProgrammingError) (1007, "Can't create database 'test'; database exists")
[SQL: -- if we create database directly, we will got a error 'Can't create database 'test'; database exists'
create schema `test`]
(Background on this error at: http://sqlalche.me/e/f405)
```

```
%%sql
-- so we can use 'if not exists' to help us
create schema if not exists `test`
```

```
 * mysql+pymysql://root:***@localhost/mydb
1 rows affected.
```

```
[]
```

Drop database

```
%%sql
drop schema `test`
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
```

```
[]
```

Use database

```
%%sql
use `test`
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
```

```
[]
```

1.2 Create tables

Statement:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...)
```

```
%%sql
create table if not exists `test`.`test_table`(
    `idUser` INT NOT NULL AUTO_INCREMENT,
    `userName` VARCHAR(45) NOT NULL,
    `password` VARCHAR(45) NOT NULL,
    `role` VARCHAR(45) NOT NULL,
    PRIMARY KEY (`idUser`))
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
```

```
[]
```

```
%%sql
select * from test_table
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
```

idUser	userName	password	role

2. Use DB, create tables with table options (NULL, Key, etc.), data types, insert, index and set

Abstract example:

```
-- Use DB
USE `mydb` ;

-- Create tables with table options (NULL, Key, etc.),data types
CREATE TABLE IF NOT EXISTS `mydb`.`User` (
  `idUser` INT NOT NULL AUTO_INCREMENT,
  `userName` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `role` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idUser`))

-- Insert
INSERT INTO `mydb`.`User` (`idUser`, `userName`, `password`, `role`) VALUES ('26', 'user_26', 'password_26',
'Guider');

-- Index
-- Enables use of statements that create or drop (remove) indexes. INDEX applies to existing tables. If you have the
CREATE privilege for a table, you can include index definitions in the CREATE TABLE statement.
CREATE INDEX indexName ON mytable(username(length));
ALTER table tableName ADD INDEX indexName(columnName);
CREATE TABLE mytable(
ID INT NOT NULL,
username VARCHAR(16) NOT NULL,
INDEX [indexName] (username(length))
);
DROP INDEX [indexName] ON mytable;

-- Set
-- A SET column can have a maximum of 64 distinct members.
CREATE TABLE myset (col SET('a', 'b', 'c', 'd'));

INSERT INTO myset (col) VALUES
('a,d'), ('d,a'), ('a,d,a'), ('a,d,d'), ('d,a,d');

SELECT col FROM myset;
+------+
| col  |
+------+
| a,d  |
| a,d  |
| a,d  |
| a,d  |
| a,d  |
+------+
5 rows in set (0.04 sec)
```

2.1 Use DB

```
%%sql
use `test`
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.




[]
```

2.2 Create tables with table options (NULL, Key, etc.), data types

Statement:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (column_name column_type)
```

Creata a table with columns and data types

```
%%sql
CREATE TABLE IF NOT EXISTS `test_tbl`(
   `test_id` INT UNSIGNED AUTO_INCREMENT,
   `test_title` VARCHAR(100) NOT NULL,
   `test_author` VARCHAR(40) NOT NULL,
   `submission_date` DATE,
   PRIMARY KEY ( `test_id` )
)ENGINE=InnoDB DEFAULT CHARSET=UTF8MB4;
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
```

```
[]
```

- If you don't want the field to be NULL, you can set the field's attribute to NOT NULL. If you enter the field's data as NULL when operating the database, you will get an error.
- AUTO_INCREMENT defines the column as a self-increasing attribute, which is generally used for the primary key, and the value will automatically increase by 1.
- The PRIMARY KEY keyword is used to define the column as the primary key. You can use multiple columns to define the primary key, separated by commas.
- ENGINE sets the storage engine, and CHARSET sets the encoding.

```
%%sql
select * from `test_tbl`
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
```

test_id **test_title** **test_author** **submission_date**

Insert data into table

Statement:

```
INSERT INTO table_name ( field1, field2,...fieldN )
                       VALUES
                       ( value1, value2,...valueN );
```

2.3 Insertion

Let's insert one row into test_tbl table

```
%%sql
insert into `test_tbl` (test_id,test_title,test_author,submission_date)
values (1,'title','author','2020-04-14')
```

```
 * mysql+pymysql://root:***@localhost/mydb
1 rows affected.
```

```
[]
```

```
%%sql
select * from `test_tbl`
```

```
 * mysql+pymysql://root:***@localhost/mydb
1 rows affected.
```

test_id	test_title	test_author	submission_date
1	title	author	2020-04-14

2.4 Index

Create index

```
CREATE INDEX indexName ON mytable(columnName(length));
```

> If it is CHAR, VARCHAR type, length can be less than the actual length of the field; if it is BLOB and TEXT types, length must be specified.

Change table structure to add index

```
ALTER table tableName ADD INDEX indexName(columnName)
```

Create index while creating tables

```
CREATE TABLE mytable(
ID INT NOT NULL,
username VARCHAR(16) NOT NULL,
INDEX [indexName] (columnName(length))
);
```

Delete index

```
DROP INDEX [indexName] ON mytable;
```

```
%%sql
create index test_index on `test_tbl`(test_id);
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
```

```
[]
```

```
%%sql
show index from `test_tbl`
```

```
 * mysql+pymysql://root:***@localhost/mydb
2 rows affected.
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_
test_tbl	0	PRIMARY	1	test_id	A	1	None	None		BTREE		
test_tbl	1	test_index	1	test_id	A	1	None	None		BTREE		

```
%%sql
drop index test_index on `test_tbl`
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
```

```
[]
```

```
%%sql
show index from `test_tbl`
```

```
 * mysql+pymysql://root:***@localhost/mydb
1 rows affected.
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_
test_tbl	0	PRIMARY	1	test_id	A	1	None	None		BTREE		

2.5 Set

A SET is a string object that can have zero or more values, each of which must be chosen from a list of permitted values specified when the table is created. SET column values that consist of multiple set members are specified with members separated by commas (,). A consequence of this is that SET member values should not themselves contain commas.

```
%%sql
create table `myset` (col set('a','b','c','d'));
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
```

```
[]
```

```
%%sql
insert into myset (col) values
('a,d'), ('d,a'), ('a,d,a'), ('a,d,d'), ('d,a,d');
```

```
 * mysql+pymysql://root:***@localhost/mydb
5 rows affected.
```

```
[]
```

```
%%sql
select col from myset;
```

```
 * mysql+pymysql://root:***@localhost/mydb
5 rows affected.
```

col

a,d
a,d
a,d
a,d
a,d

3. Select – From – Where – Group By – Having - Order By – Limit

Abstract statement

```
-- select-from.
-- This statement can allow you to extract some particular columns from some particular tables
SELECT field1, field2,...fieldN FROM table_name1, table_name2...

-- select-from-where
-- This statement can allow you to extract some data based on particular condition
SELECT field1, field2,...fieldN FROM table_name1, table_name2...
[WHERE condition1 [AND [OR]] condition2.....

-- select-from-where-group by
-- The GROUP BY statement groups the result set based on one or more columns.
SELECT column_name, function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name;

-- having
-- The reason for adding the HAVING clause in SQL is that the WHERE keyword cannot be used with aggregate functions.
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
HAVING aggregate_function(column_name) operator value;

-- order by
-- The ORDER BY keyword is used to sort the result set by one column or multiple columns.
SELECT column_name,column_name
FROM table_name
ORDER BY column_name,column_name ASC|DESC;

-- limit
-- The LIMIT clause is used to limit the amount of data returned by the SELECT statement.
SELECT column1, column2, columnN
FROM table_name
LIMIT [no of rows]
```

```
%%sql
use `mydb`
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
```

```
[]
```

```
%%sql
show tables
```

```
 * mysql+pymysql://root:***@localhost/mydb
9 rows affected.
```

Tables_in_mydb

Admin
Customer
CustomerOrder
Flight
Guider
Room
Supplier
TravelProduct
User

3.1 select-from

```
%%sql
-- '*' means selec all columns from tables
select * from User;
```

```
 * mysql+pymysql://root:***@localhost/mydb
25 rows affected.
```

idUser	userName	password	role
1	user_1	password_1	Customer
2	user_2	password_2	Customer
3	user_3	password_3	Customer
4	user_4	password_4	Customer
5	user_5	password_5	Customer
6	user_6	password_6	Customer
7	user_7	password_7	Customer
8	user_8	password_8	Customer
9	user_9	password_9	Customer
10	user_10	password_10	Customer
11	user_11	password_11	Admin
12	user_12	password_12	Admin
13	user_13	password_13	Admin
14	user_14	password_14	Admin
15	user_15	password_15	Admin
16	user_16	password_16	Supplier
17	user_17	password_17	Supplier
18	user_18	password_18	Supplier
19	user_19	password_19	Supplier
20	user_20	password_20	supplier
21	user_21	password_21	Guider
22	user_22	password_22	Guider
23	user_23	password_23	Guider
24	user_24	password_24	Guider
25	user_25	password_25	Guider

3.2 select-from-where

```

```

```
%%sql
select * from User
where idUser = 2;
```

```
 * mysql+pymysql://root:***@localhost/mydb
1 rows affected.
```

idUser	userName	password	role
2	user_2	password_2	Customer

3.3 select-from-where-group by

```
%%sql
select role,count(*) from User
where idUser < 21
group by role
```

```
 * mysql+pymysql://root:***@localhost/mydb
3 rows affected.
```

role	count(*)
Customer	10
Admin	5
Supplier	5

3.4 having

```
%%sql
select role,count(*) number from User
where idUser < 21
group by role
having number < 10
```

```
 * mysql+pymysql://root:***@localhost/mydb
2 rows affected.
```

role	number
Admin	5
Supplier	5

3.5 order by

```
%%sql
select * from User
order by idUser desc
```

```
 * mysql+pymysql://root:***@localhost/mydb
25 rows affected.
```

idUser	userName	password	role
25	user_25	password_25	Guider
24	user_24	password_24	Guider
23	user_23	password_23	Guider

22	user_22	password_22	Guider
21	user_21	password_21	Guider
20	user_20	password_20	supplier
19	user_19	password_19	Supplier
18	user_18	password_18	Supplier
17	user_17	password_17	Supplier
16	user_16	password_16	Supplier
15	user_15	password_15	Admin
14	user_14	password_14	Admin
13	user_13	password_13	Admin
12	user_12	password_12	Admin
11	user_11	password_11	Admin
10	user_10	password_10	Customer
9	user_9	password_9	Customer
8	user_8	password_8	Customer
7	user_7	password_7	Customer
6	user_6	password_6	Customer
5	user_5	password_5	Customer
4	user_4	password_4	Customer
3	user_3	password_3	Customer
2	user_2	password_2	Customer
1	user_1	password_1	Customer

3.6 limit

```
%%sql
select * from User
limit 5
```

```
 * mysql+pymysql://root:***@localhost/mydb
5 rows affected.
```

idUser	userName	password	role
1	user_1	password_1	Customer
2	user_2	password_2	Customer
3	user_3	password_3	Customer
4	user_4	password_4	Customer
5	user_5	password_5	Customer

4. Select 10 functions in SQL statement of your choice

(including but not limited to Comparison, control flow, cast, string, numeric, date/time manipulation, calculation, conversion, aggregate functions, etc.)

4.1 Comparison

Operator	Function
=	Equal to
<=>	Safe equal
<>	Or! = Not equal
<=	Less than or equal to
>=	Greater than or equal to

Operator	Function
>	Greater than
IS NULL or ISNULL	to determine whether a value is empty
IS NOT NULL	judge whether a value is not empty
BETWEEN AND	determines whether a value falls between two values

Greater >

```
%%sql
select * from TravelProduct
where platformProfit > 200
```

```
 * mysql+pymysql://root:***@localhost/mydb
20 rows affected.
```

idTravelProduct	idFlight	idRoom	idGuider	description	platformProfit
1	19	30	25	Good	628
2	12	22	21	Mid	449
3	20	21	25	Good	658
4	1	7	25	Mid	761
5	6	15	25	Good	520
6	17	29	25	Perfect	437
7	14	28	25	Mid	633
8	22	14	23	Mid	416
9	23	15	25	Good	518
10	5	13	25	Perfect	483
11	20	40	23	Perfect	338
12	23	19	23	Good	431
13	1	7	22	Mid	432
14	2	15	23	Perfect	676
15	23	21	21	Good	452
16	19	19	24	Mid	552
17	9	7	22	Good	717
18	5	15	25	Mid	307
19	18	30	23	Mid	662
20	2	41	25	Mid	425

between and comparison operator

```
%%sql
select * from TravelProduct
where platformProfit between 300 and 400
```

```
 * mysql+pymysql://root:***@localhost/mydb
2 rows affected.
```

idTravelProduct	idFlight	idRoom	idGuider	description	platformProfit
11	20	40	23	Perfect	338
18	5	15	25	Mid	307

4.2 Control flow

There are four Control Flow functions in MySQL - CASE operator, IF/ELSE construct, IFNULL, and NULLIF.

4.2.1 CASE

Version 1:

```
CASE value WHEN [compare_value] THEN result [WHEN [compare_value] THEN result ...] [ELSE result] END
```

In this version, result is returned when value is equal to compare_value. If nothing is matched, the result after ELSE is returned, or NULL is returned if there is no ELSE part.

Version 2:

```
CASE WHEN [condition] THEN result [WHEN [condition] THEN result ...] [ELSE result] END
```

In this version, if condition is true, result is returned. If nothing is matched, the result after ELSE is returned, or NULL is returned if there is no ELSE part.

Let's make a summary to check the number of each interval of the travel product profit.

```
%%sql
select
    case
        when platformProfit between 1 and 300
            then '(0,300]'
        when platformProfit between 301 and 600
            then '(300,600]'
        when platformProfit between 601 and 900
            then '(600,900]'
        else '(900,)'
    end as profit_interval,
    count(idTravelProduct) product_number
from TravelProduct
group by profit_interval
order by profit_interval
```

```
 * mysql+pymysql://root:***@localhost/mydb
2 rows affected.
```

profit_interval	product_number
(300,600]	13
(600,900]	7

4.2.2 IF/ELSE

```
%%sql
select idTravelProduct, platformProfit, if(platformProfit > 500, 'high profit', 'low profit') profit_type
from TravelProduct
```

```
 * mysql+pymysql://root:***@localhost/mydb
20 rows affected.
```

idTravelProduct	platformProfit	profit_type
1	628	high profit
2	449	low profit
3	658	high profit
4	761	high profit
5	520	high profit
6	437	low profit
7	633	high profit
8	416	low profit

9	518	high profit
10	483	low profit
11	338	low profit
12	431	low profit
13	432	low profit
14	676	high profit
15	452	low profit
16	552	high profit
17	717	high profit
18	307	low profit
19	662	high profit
20	425	low profit

4.2.3 IFNULL

```
IFNULL(expr1,expr2)
```

If expr1 is not NULL, the function returns expr1. Otherwise it returns expr2.

4.2.4 NULLIF

```
NULLIF(expr1,expr2)
```

Returns NULL if expr1 = expr2 is true, otherwise returns expr1.

5. Cast

The CAST function converts any type of value to a value of the specified type

```
CAST(expression AS TYPE);
```

The CAST () function converts any type of value to a value of the specified type.

The target type can be one of the following types:

- BINARY
- CHAR
- DATE
- DATETIME
- TIME
- DECIMAL
- SIGNED
- UNSIGNED

Let's add 'price' as prefix of platformProfit

Here we can use concat function to concat two string

```
%%sql
show columns from TravelProduct
```

```
 * mysql+pymysql://root:***@localhost/mydb
6 rows affected.
```

Field	Type	Null	Key	Default	Extra
idTravelProduct	int	NO	PRI	None	auto_increment
idFlight	int	NO	MUL	None	

idRoom	int	NO	MUL	None
idGuider	int	NO	MUL	None
description	varchar(45)	NO		None
platformProfit	decimal(45,0)	NO		None

The 'platformProfit' is `decimal`. So we can change the type as `char` first

```
%%sql
select concat('price ',cast(platformProfit as char)) as profit
from TravelProduct
```

```
 * mysql+pymysql://root:***@localhost/mydb
20 rows affected.
```

profit
price 628
price 449
price 658
price 761
price 520
price 437
price 633
price 416
price 518
price 483
price 338
price 431
price 432
price 676
price 452
price 552
price 717
price 307
price 662
price 425

6. String

String functions in MySQL

Function	Description
ASCII	Returns the ASCII value for the specific character
CHAR_LENGTH	Returns the length of a string (in characters)
CHARACTER_LENGTH	Returns the length of a string (in characters)
CONCAT	Adds two or more expressions together
CONCAT_WS	Adds two or more expressions together with a separator
FIELD	Returns the index position of a value in a list of values
FIND_IN_SET	Returns the position of a string within a list of strings
FORMAT	Formats a number to a format like "#,###,###.##", rounded to a specified number of decimal places
INSERT	Inserts a string within a string at the specified position and for a certain number of characters

Function	Description
INSTR	Returns the position of the first occurrence of a string in another string
LCASE	Converts a string to lower-case
LEFT	Extracts a number of characters from a string (starting from left)
LENGTH	Returns the length of a string (in bytes)
LOCATE	Returns the position of the first occurrence of a substring in a string
LOWER	Converts a string to lower-case
LPAD	Left-pads a string with another string, to a certain length
LTRIM	Removes leading spaces from a string
MID	Extracts a substring from a string (starting at any position)
POSITION	Returns the position of the first occurrence of a substring in a string
REPEAT	Repeats a string as many times as specified
REPLACE	Replaces all occurrences of a substring within a string, with a new substring
REVERSE	Reverses a string and returns the result
RIGHT	Extracts a number of characters from a string (starting from right)
RPAD	Right-pads a string with another string, to a certain length
RTRIM	Removes trailing spaces from a string
SPACE	Returns a string of the specified number of space characters
STRCMP	Compares two strings
SUBSTR	Extracts a substring from a string (starting at any position)
SUBSTRING	Extracts a substring from a string (starting at any position)
SUBSTRING_INDEX	Returns a substring of a string before a specified number of delimiter occurs
TRIM	Removes leading and trailing spaces from a string
UCASE	Converts a string to upper-case
UPPER	Converts a string to upper-case

```
%%sql
select ucase('test')
```

```
 * mysql+pymysql://root:***@localhost/mydb
1 rows affected.
```

ucase('test')
TEST

```
%%sql
select format(platformProfit,2)
from TravelProduct
```

```
 * mysql+pymysql://root:***@localhost/mydb
20 rows affected.
```

format(platformProfit,2)
628.00
449.00
658.00
761.00
520.00

437.00
633.00
416.00
518.00
483.00
338.00
431.00
432.00
676.00
452.00
552.00
717.00
307.00
662.00
425.00

7. Numeric

Numberic functions in MySQL

Function	Description
ABS	Returns the absolute value of a number
ACOS	Returns the arc cosine of a number
ASIN	Returns the arc sine of a number
ATAN	Returns the arc tangent of one or two numbers
ATAN2	Returns the arc tangent of two numbers
AVG	Returns the average value of an expression
CEIL	Returns the smallest integer value that is >= to a number
CEILING	Returns the smallest integer value that is >= to a number
COS	Returns the cosine of a number
COT	Returns the cotangent of a number
COUNT	Returns the number of records returned by a select query
DEGREES	Converts a value in radians to degrees
DIV	Used for integer division
EXP	Returns e raised to the power of a specified number
FLOOR	Returns the largest integer value that is <= to a number
GREATEST	Returns the greatest value of the list of arguments
LEAST	Returns the smallest value of the list of arguments
LN	Returns the natural logarithm of a number
LOG	Returns the natural logarithm of a number, or the logarithm of a number to a specified base
LOG10	Returns the natural logarithm of a number to base 10
LOG2	Returns the natural logarithm of a number to base 2
MAX	Returns the maximum value in a set of values
MIN	Returns the minimum value in a set of values
MOD	Returns the remainder of a number divided by another number
PI	Returns the value of PI
POW	Returns the value of a number raised to the power of another number
POWER	Returns the value of a number raised to the power of another number
RADIANS	Converts a degree value into radians

Function	Description
RAND	Returns a random number
ROUND	Rounds a number to a specified number of decimal places
SIGN	Returns the sign of a number
SIN	Returns the sine of a number
SQRT	Returns the square root of a number
SUM	Calculates the sum of a set of values
TAN	Returns the tangent of a number
TRUNCATE	Truncates a number to the specified number of decimal places

Let's check the profit that we sold out all products

```
%%sql
select sum(platformProfit)
from TravelProduct
```

```
 * mysql+pymysql://root:***@localhost/mydb
1 rows affected.
```

sum(platformProfit)
10495

Let's chech the average profit

```
%%sql
select sum(platformProfit)/count(*)
from TravelProduct
```

```
 * mysql+pymysql://root:***@localhost/mydb
1 rows affected.
```

sum(platformProfit)/count(*)
524.7500

```
%%sql
select avg(platformProfit)
from TravelProduct
```

```
 * mysql+pymysql://root:***@localhost/mydb
1 rows affected.
```

avg(platformProfit)
524.7500

And we can do `round` to it

```
%%sql
select round(avg(platformProfit))
from TravelProduct
```

```
 * mysql+pymysql://root:***@localhost/mydb
1 rows affected.
```

round(avg(platformProfit))
525

8. Date/time manipulation

Function	Description
ADDDATE	Adds a time/date interval to a date and then returns the date
ADDTIME	Adds a time interval to a time/datetime and then returns the time/datetime
CURDATE	Returns the current date
CURRENT_DATE	Returns the current date
CURRENT_TIME	Returns the current time
CURRENT_TIMESTAMP	Returns the current date and time
CURTIME	Returns the current time
DATE	Extracts the date part from a datetime expression
DATEDIFF	Returns the number of days between two date values
DATE_ADD	Adds a time/date interval to a date and then returns the date
DATE_FORMAT	Formats a date
DATE_SUB	Subtracts a time/date interval from a date and then returns the date
DAY	Returns the day of the month for a given date
DAYNAME	Returns the weekday name for a given date
DAYOFMONTH	Returns the day of the month for a given date
DAYOFWEEK	Returns the weekday index for a given date
DAYOFYEAR	Returns the day of the year for a given date
EXTRACT	Extracts a part from a given date
FROM_DAYS	Returns a date from a numeric datevalue
HOUR	Returns the hour part for a given date
LAST_DAY	Extracts the last day of the month for a given date
LOCALTIME	Returns the current date and time
LOCALTIMESTAMP	Returns the current date and time
MAKEDATE	Creates and returns a date based on a year and a number of days value
MAKETIME	Creates and returns a time based on an hour, minute, and second value
MICROSECOND	Returns the microsecond part of a time/datetime
MINUTE	Returns the minute part of a time/datetime
MONTH	Returns the month part for a given date
MONTHNAME	Returns the name of the month for a given date
NOW	Returns the current date and time
PERIOD_ADD	Adds a specified number of months to a period
PERIOD_DIFF	Returns the difference between two periods
QUARTER	Returns the quarter of the year for a given date value
SECOND	Returns the seconds part of a time/datetime
SEC_TO_TIME	Returns a time value based on the specified seconds
STR_TO_DATE	Returns a date based on a string and a format
SUBDATE	Subtracts a time/date interval from a date and then returns the date
SUBTIME	Subtracts a time interval from a datetime and then returns the time/datetime
SYSDATE	Returns the current date and time
TIME	Extracts the time part from a given time/datetime
TIME_FORMAT	Formats a time by a specified format
TIME_TO_SEC	Converts a time value into seconds

Function	Description
TIMEDIFF	Returns the difference between two time/datetime expressions
TIMESTAMP	Returns a datetime value based on a date or datetime value
TO_DAYS	Returns the number of days between a date and date "0000-00-00"
WEEK	Returns the week number for a given date
WEEKDAY	Returns the weekday number for a given date
WEEKOFYEAR	Returns the week number for a given date
YEAR	Returns the year part for a given date
YEARWEEK	Returns the year and week number for a given date

```
%%sql
select *, month(roomDate), dayofmonth(roomDate) from room
```

```
 * mysql+pymysql://root:***@localhost/mydb
21 rows affected.
```

idRoom	idSupplier	roomType	roomDate	price	ifAvailable	month(roomDate)	dayofmonth(roomDate)
7	16	single	2020-05-09	341	1	5	9
13	16	double	2020-06-13	281	1	6	13
14	17	single	2020-05-09	241	1	5	9
15	18	double	2020-05-05	485	1	5	5
16	19	single	2020-05-05	382	1	5	5
17	20	double	2020-04-16	131	1	4	16
18	16	single	2020-04-24	402	1	4	24
19	17	single	2020-06-21	144	1	6	21
20	18	single	2020-05-25	345	1	5	25
21	19	single	2020-05-17	487	1	5	17
22	20	single	2020-06-26	151	1	6	26
28	16	double	2020-04-29	459	1	4	29
29	17	single	2020-05-13	109	1	5	13
30	18	double	2020-05-11	288	1	5	11
31	19	single	2020-04-03	409	1	4	3
32	20	single	2020-05-23	307	1	5	23
38	16	single	2020-06-18	338	1	6	18
39	17	double	2020-06-25	215	1	6	25
40	18	double	2020-04-15	135	1	4	15
41	19	single	2020-06-05	274	1	6	5
42	20	double	2020-06-06	307	1	6	6

9. Calculation

When data is retrieved from a MySQL database it is not always in the form we need it. For example, we may need to display a customer name and address as a single text string, but in reality the name and address are held in separate tables.

```
%%sql
select * from room
```

```
 * mysql+pymysql://root:***@localhost/mydb
21 rows affected.
```

idRoom	idSupplier	roomType	roomDate	price	ifAvailable
7	16	single	2020-05-09	341	1
13	16	double	2020-06-13	281	1
14	17	single	2020-05-09	241	1
15	18	double	2020-05-05	485	1
16	19	single	2020-05-05	382	1
17	20	double	2020-04-16	131	1
18	16	single	2020-04-24	402	1
19	17	single	2020-06-21	144	1
20	18	single	2020-05-25	345	1
21	19	single	2020-05-17	487	1
22	20	single	2020-06-26	151	1
28	16	double	2020-04-29	459	1
29	17	single	2020-05-13	109	1
30	18	double	2020-05-11	288	1
31	19	single	2020-04-03	409	1
32	20	single	2020-05-23	307	1
38	16	single	2020-06-18	338	1
39	17	double	2020-06-25	215	1
40	18	double	2020-04-15	135	1
41	19	single	2020-06-05	274	1
42	20	double	2020-06-06	307	1

Let's say that for now we need to reduce 10 for any user that order a room. So let's check the price after using cupons

```
%%sql
select *, price - 10 cupons
from room
```

```
 * mysql+pymysql://root:***@localhost/mydb
21 rows affected.
```

idRoom	idSupplier	roomType	roomDate	price	ifAvailable	cupons
7	16	single	2020-05-09	341	1	331
13	16	double	2020-06-13	281	1	271
14	17	single	2020-05-09	241	1	231
15	18	double	2020-05-05	485	1	475
16	19	single	2020-05-05	382	1	372
17	20	double	2020-04-16	131	1	121
18	16	single	2020-04-24	402	1	392
19	17	single	2020-06-21	144	1	134
20	18	single	2020-05-25	345	1	335
21	19	single	2020-05-17	487	1	477
22	20	single	2020-06-26	151	1	141
28	16	double	2020-04-29	459	1	449
29	17	single	2020-05-13	109	1	99
30	18	double	2020-05-11	288	1	278
31	19	single	2020-04-03	409	1	399

32	20	single	2020-05-23	307	1	297
38	16	single	2020-06-18	338	1	328
39	17	double	2020-06-25	215	1	205
40	18	double	2020-04-15	135	1	125
41	19	single	2020-06-05	274	1	264
42	20	double	2020-06-06	307	1	297

10. Conversion

MySQL Conversion Functions convert a value from one data type to another. Conversions can be conducted between string, date, and numeric type of data. There are three Conversion Functions in MySQL: CONVERT, CAST, BINARY.

10.1 CONVERT

CONVERT can be used in either of the following two forms:

Form 1:

```
CONVERT(expr,type)
```

In this form, CONVERT takes a value in the form of expr and converts it to a value of type.

- BINARY
- CHAR
- DATE
- DATETIME
- TIME
- DECIMAL
- SIGNED
- UNSIGNED

```
%%sql
SELECT CONVERT('1998-03-16 00:00:00', DATE) AS date1,
       CONVERT('98-03-16 00:00:00', DATE) AS date2,
       CONVERT('19980316', DATE) AS date3,
       CONVERT('980316', DATE) AS date4;
```

```
 * mysql+pymysql://root:***@localhost/mydb
1 rows affected.
```

date1	date2	date3	date4
1998-03-16	1998-03-16	1998-03-16	1998-03-16

Form 2:

```
CONVERT(expr USING transcoding_name)
```

In this form, CONVERT() with USING is used to convert data between different character sets (utf8, latin1, ...) In MySQL, transcoding names are the same as the corresponding character set names.

```
%%sql
SELECT CONVERT('abc' USING UTF8MB4);
```

```
 * mysql+pymysql://root:***@localhost/mydb
1 rows affected.
```

CONVERT('abc' USING UTF8MB4)
abc

10.2 CAST

The CAST function converts any type of value to a value of the specified type

```
CAST(expression AS TYPE);
```

The CAST () function converts any type of value to a value of the specified type.

The target type can be one of the following types:

- BINARY
- CHAR
- DATE
- DATETIME
- TIME
- DECIMAL
- SIGNED
- UNSIGNED

```
%%sql
select concat('price ',cast(platformProfit as char)) as profit
from TravelProduct
```

```
 * mysql+pymysql://root:***@localhost/mydb
20 rows affected.
```

profit
price 628
price 449
price 658
price 761
price 520
price 437
price 633
price 416
price 518
price 483
price 338
price 431
price 432
price 676
price 452
price 552
price 717
price 307
price 662
price 425

10.3 BINARY

BINARY str BINARY is an operator rather than function. It casts a string following it to a binary string.

BINARY str is shorthand for CAST(str AS BINARY) or CONVERT(expr, BINARY)

11. Aggregate functions

An aggregate function performs a calculation on multiple values and returns a single value.

Aggregate function	Description
AVG()	Return the average of non-NULL values.
BIT_AND()	Return bitwise AND.
BIT_OR()	Return bitwise OR.
BIT_XOR()	Return bitwise XOR.
COUNT()	Return the number of rows in a group, including rows with NULL values.
GROUP_CONCAT()	Return a concatenated string.
JSON_ARRAYAGG()	Return result set as a single JSON array.
JSON_OBJECTAGG()	Return result set as a single JSON object.
MAX()	Return the highest value (maximum) in a set of non-NULL values.
MIN()	Return the lowest value (minimum) in a set of non-NULL values.
STDEV()	Return the population standard deviation.
STDDEV_POP()	Return the population standard deviation.
STDDEV_SAMP()	Return the sample standard deviation.
SUM()	Return the summation of all non-NULL values a set.
VAR_POP()	Return the population standard variance.
VARP_SAM()	Return the sample variance.
VARIANCE()	Return the population standard variance.

Let's check the total profit for different interval

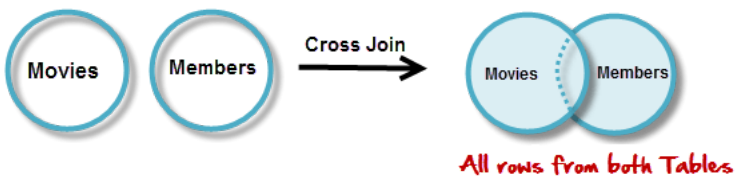
```
%%sql
select sum(platformProfit) total_profit,
    case
        when platformProfit between 1 and 300
            then '(0,300]'
        when platformProfit between 301 and 600
            then '(300,600]'
        when platformProfit between 601 and 900
            then '(600,900]'
        else '(900,)'
    end as profit_interval,
    count(idTravelProduct) product_number
from TravelProduct
group by profit_interval
order by profit_interval
```

```
 * mysql+pymysql://root:***@localhost/mydb
2 rows affected.
```

total_profit	profit_interval	product_number
5760	(300,600]	13
4735	(600,900]	7

5. Perform JOIN operation

(including INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, CROSS JOIN operations and display the sample result using SELECT statement)




```
%%sql
select * from  Customer c
cross join User u
```

```
 * mysql+pymysql://root:***@localhost/mydb
250 rows affected.
```

idCustomer	idUser	idUser_1	userName	password	role
1	1	1	user_1	password_1	Customer
2	2	1	user_1	password_1	Customer
3	3	1	user_1	password_1	Customer
4	4	1	user_1	password_1	Customer
5	5	1	user_1	password_1	Customer
6	6	1	user_1	password_1	Customer
7	7	1	user_1	password_1	Customer
8	8	1	user_1	password_1	Customer
9	9	1	user_1	password_1	Customer
10	10	1	user_1	password_1	Customer
1	1	2	user_2	password_2	Customer
2	2	2	user_2	password_2	Customer
3	3	2	user_2	password_2	Customer
4	4	2	user_2	password_2	Customer
5	5	2	user_2	password_2	Customer
6	6	2	user_2	password_2	Customer
7	7	2	user_2	password_2	Customer
8	8	2	user_2	password_2	Customer
9	9	2	user_2	password_2	Customer
10	10	2	user_2	password_2	Customer
1	1	3	user_3	password_3	Customer
2	2	3	user_3	password_3	Customer
3	3	3	user_3	password_3	Customer
4	4	3	user_3	password_3	Customer
5	5	3	user_3	password_3	Customer
6	6	3	user_3	password_3	Customer
7	7	3	user_3	password_3	Customer
8	8	3	user_3	password_3	Customer
9	9	3	user_3	password_3	Customer
10	10	3	user_3	password_3	Customer
1	1	4	user_4	password_4	Customer
2	2	4	user_4	password_4	Customer
3	3	4	user_4	password_4	Customer
4	4	4	user_4	password_4	Customer
5	5	4	user_4	password_4	Customer
6	6	4	user_4	password_4	Customer
7	7	4	user_4	password_4	Customer
8	8	4	user_4	password_4	Customer
9	9	4	user_4	password_4	Customer
10	10	4	user_4	password_4	Customer
1	1	5	user_5	password_5	Customer

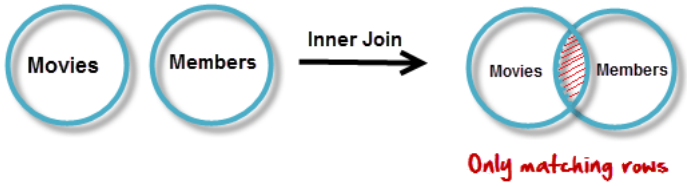
2	2	5	user_5	password_5	Customer
3	3	5	user_5	password_5	Customer
4	4	5	user_5	password_5	Customer
5	5	5	user_5	password_5	Customer
6	6	5	user_5	password_5	Customer
7	7	5	user_5	password_5	Customer
8	8	5	user_5	password_5	Customer
9	9	5	user_5	password_5	Customer
10	10	5	user_5	password_5	Customer
1	1	6	user_6	password_6	Customer
2	2	6	user_6	password_6	Customer
3	3	6	user_6	password_6	Customer
4	4	6	user_6	password_6	Customer
5	5	6	user_6	password_6	Customer
6	6	6	user_6	password_6	Customer
7	7	6	user_6	password_6	Customer
8	8	6	user_6	password_6	Customer
9	9	6	user_6	password_6	Customer
10	10	6	user_6	password_6	Customer
1	1	7	user_7	password_7	Customer
2	2	7	user_7	password_7	Customer
3	3	7	user_7	password_7	Customer
4	4	7	user_7	password_7	Customer
5	5	7	user_7	password_7	Customer
6	6	7	user_7	password_7	Customer
7	7	7	user_7	password_7	Customer
8	8	7	user_7	password_7	Customer
9	9	7	user_7	password_7	Customer
10	10	7	user_7	password_7	Customer
1	1	8	user_8	password_8	Customer
2	2	8	user_8	password_8	Customer
3	3	8	user_8	password_8	Customer
4	4	8	user_8	password_8	Customer
5	5	8	user_8	password_8	Customer
6	6	8	user_8	password_8	Customer
7	7	8	user_8	password_8	Customer
8	8	8	user_8	password_8	Customer
9	9	8	user_8	password_8	Customer
10	10	8	user_8	password_8	Customer
1	1	9	user_9	password_9	Customer
2	2	9	user_9	password_9	Customer
3	3	9	user_9	password_9	Customer
4	4	9	user_9	password_9	Customer
5	5	9	user_9	password_9	Customer
6	6	9	user_9	password_9	Customer
7	7	9	user_9	password_9	Customer
8	8	9	user_9	password_9	Customer

9	9	9	user_9	password_9	Customer
10	10	9	user_9	password_9	Customer
1	1	10	user_10	password_10	Customer
2	2	10	user_10	password_10	Customer
3	3	10	user_10	password_10	Customer
4	4	10	user_10	password_10	Customer
5	5	10	user_10	password_10	Customer
6	6	10	user_10	password_10	Customer
7	7	10	user_10	password_10	Customer
8	8	10	user_10	password_10	Customer
9	9	10	user_10	password_10	Customer
10	10	10	user_10	password_10	Customer
1	1	11	user_11	password_11	Admin
2	2	11	user_11	password_11	Admin
3	3	11	user_11	password_11	Admin
4	4	11	user_11	password_11	Admin
5	5	11	user_11	password_11	Admin
6	6	11	user_11	password_11	Admin
7	7	11	user_11	password_11	Admin
8	8	11	user_11	password_11	Admin
9	9	11	user_11	password_11	Admin
10	10	11	user_11	password_11	Admin
1	1	12	user_12	password_12	Admin
2	2	12	user_12	password_12	Admin
3	3	12	user_12	password_12	Admin
4	4	12	user_12	password_12	Admin
5	5	12	user_12	password_12	Admin
6	6	12	user_12	password_12	Admin
7	7	12	user_12	password_12	Admin
8	8	12	user_12	password_12	Admin
9	9	12	user_12	password_12	Admin
10	10	12	user_12	password_12	Admin
1	1	13	user_13	password_13	Admin
2	2	13	user_13	password_13	Admin
3	3	13	user_13	password_13	Admin
4	4	13	user_13	password_13	Admin
5	5	13	user_13	password_13	Admin
6	6	13	user_13	password_13	Admin
7	7	13	user_13	password_13	Admin
8	8	13	user_13	password_13	Admin
9	9	13	user_13	password_13	Admin
10	10	13	user_13	password_13	Admin
1	1	14	user_14	password_14	Admin
2	2	14	user_14	password_14	Admin
3	3	14	user_14	password_14	Admin
4	4	14	user_14	password_14	Admin

5	5	14	user_14	password_14	Admin
6	6	14	user_14	password_14	Admin
7	7	14	user_14	password_14	Admin
8	8	14	user_14	password_14	Admin
9	9	14	user_14	password_14	Admin
10	10	14	user_14	password_14	Admin
1	1	15	user_15	password_15	Admin
2	2	15	user_15	password_15	Admin
3	3	15	user_15	password_15	Admin
4	4	15	user_15	password_15	Admin
5	5	15	user_15	password_15	Admin
6	6	15	user_15	password_15	Admin
7	7	15	user_15	password_15	Admin
8	8	15	user_15	password_15	Admin
9	9	15	user_15	password_15	Admin
10	10	15	user_15	password_15	Admin
1	1	16	user_16	password_16	Supplier
2	2	16	user_16	password_16	Supplier
3	3	16	user_16	password_16	Supplier
4	4	16	user_16	password_16	Supplier
5	5	16	user_16	password_16	Supplier
6	6	16	user_16	password_16	Supplier
7	7	16	user_16	password_16	Supplier
8	8	16	user_16	password_16	Supplier
9	9	16	user_16	password_16	Supplier
10	10	16	user_16	password_16	Supplier
1	1	17	user_17	password_17	Supplier
2	2	17	user_17	password_17	Supplier
3	3	17	user_17	password_17	Supplier
4	4	17	user_17	password_17	Supplier
5	5	17	user_17	password_17	Supplier
6	6	17	user_17	password_17	Supplier
7	7	17	user_17	password_17	Supplier
8	8	17	user_17	password_17	Supplier
9	9	17	user_17	password_17	Supplier
10	10	17	user_17	password_17	Supplier
1	1	18	user_18	password_18	Supplier
2	2	18	user_18	password_18	Supplier
3	3	18	user_18	password_18	Supplier
4	4	18	user_18	password_18	Supplier
5	5	18	user_18	password_18	Supplier
6	6	18	user_18	password_18	Supplier
7	7	18	user_18	password_18	Supplier
8	8	18	user_18	password_18	Supplier
9	9	18	user_18	password_18	Supplier
10	10	18	user_18	password_18	Supplier
1	1	19	user_19	password_19	Supplier

2	2	19	user_19	password_19	Supplier
3	3	19	user_19	password_19	Supplier
4	4	19	user_19	password_19	Supplier
5	5	19	user_19	password_19	Supplier
6	6	19	user_19	password_19	Supplier
7	7	19	user_19	password_19	Supplier
8	8	19	user_19	password_19	Supplier
9	9	19	user_19	password_19	Supplier
10	10	19	user_19	password_19	Supplier
1	1	20	user_20	password_20	supplier
2	2	20	user_20	password_20	supplier
3	3	20	user_20	password_20	supplier
4	4	20	user_20	password_20	supplier
5	5	20	user_20	password_20	supplier
6	6	20	user_20	password_20	supplier
7	7	20	user_20	password_20	supplier
8	8	20	user_20	password_20	supplier
9	9	20	user_20	password_20	supplier
10	10	20	user_20	password_20	supplier
1	1	21	user_21	password_21	Guider
2	2	21	user_21	password_21	Guider
3	3	21	user_21	password_21	Guider
4	4	21	user_21	password_21	Guider
5	5	21	user_21	password_21	Guider
6	6	21	user_21	password_21	Guider
7	7	21	user_21	password_21	Guider
8	8	21	user_21	password_21	Guider
9	9	21	user_21	password_21	Guider
10	10	21	user_21	password_21	Guider
1	1	22	user_22	password_22	Guider
2	2	22	user_22	password_22	Guider
3	3	22	user_22	password_22	Guider
4	4	22	user_22	password_22	Guider
5	5	22	user_22	password_22	Guider
6	6	22	user_22	password_22	Guider
7	7	22	user_22	password_22	Guider
8	8	22	user_22	password_22	Guider
9	9	22	user_22	password_22	Guider
10	10	22	user_22	password_22	Guider
1	1	23	user_23	password_23	Guider
2	2	23	user_23	password_23	Guider
3	3	23	user_23	password_23	Guider
4	4	23	user_23	password_23	Guider
5	5	23	user_23	password_23	Guider
6	6	23	user_23	password_23	Guider
7	7	23	user_23	password_23	Guider

8	8	23	user_23	password_23	Guider
9	9	23	user_23	password_23	Guider
10	10	23	user_23	password_23	Guider
1	1	24	user_24	password_24	Guider
2	2	24	user_24	password_24	Guider
3	3	24	user_24	password_24	Guider
4	4	24	user_24	password_24	Guider
5	5	24	user_24	password_24	Guider
6	6	24	user_24	password_24	Guider
7	7	24	user_24	password_24	Guider
8	8	24	user_24	password_24	Guider
9	9	24	user_24	password_24	Guider
10	10	24	user_24	password_24	Guider
1	1	25	user_25	password_25	Guider
2	2	25	user_25	password_25	Guider
3	3	25	user_25	password_25	Guider
4	4	25	user_25	password_25	Guider
5	5	25	user_25	password_25	Guider
6	6	25	user_25	password_25	Guider
7	7	25	user_25	password_25	Guider
8	8	25	user_25	password_25	Guider
9	9	25	user_25	password_25	Guider
10	10	25	user_25	password_25	Guider




```
%%sql
select * from  Customer c
inner join User u
on c.idUser = u.idUser
```

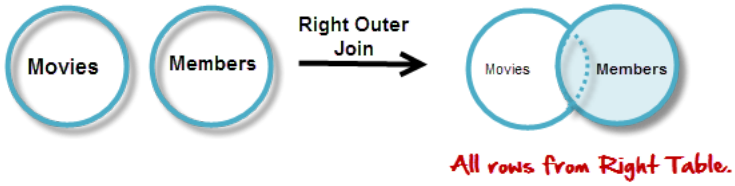
```
 * mysql+pymysql://root:***@localhost/mydb
10 rows affected.
```

idCustomer	idUser	idUser_1	userName	password	role
1	1	1	user_1	password_1	Customer
2	2	2	user_2	password_2	Customer
3	3	3	user_3	password_3	Customer
4	4	4	user_4	password_4	Customer
5	5	5	user_5	password_5	Customer
6	6	6	user_6	password_6	Customer
7	7	7	user_7	password_7	Customer
8	8	8	user_8	password_8	Customer
9	9	9	user_9	password_9	Customer
10	10	10	user_10	password_10	Customer


```
%%sql
select * from  Customer c
left join User u
on c.idUser = u.idUser
```

```
 * mysql+pymysql://root:***@localhost/mydb
10 rows affected.
```

idCustomer	idUser	idUser_1	userName	password	role
1	1	1	user_1	password_1	Customer
2	2	2	user_2	password_2	Customer
3	3	3	user_3	password_3	Customer
4	4	4	user_4	password_4	Customer
5	5	5	user_5	password_5	Customer
6	6	6	user_6	password_6	Customer
7	7	7	user_7	password_7	Customer
8	8	8	user_8	password_8	Customer
9	9	9	user_9	password_9	Customer
10	10	10	user_10	password_10	Customer




```
%%sql
select * from  Customer c
right join User u
on c.idUser = u.idUser
```

```
 * mysql+pymysql://root:***@localhost/mydb
25 rows affected.
```

idCustomer	idUser	idUser_1	userName	password	role
1	1	1	user_1	password_1	Customer
2	2	2	user_2	password_2	Customer
3	3	3	user_3	password_3	Customer
4	4	4	user_4	password_4	Customer
5	5	5	user_5	password_5	Customer
6	6	6	user_6	password_6	Customer
7	7	7	user_7	password_7	Customer
8	8	8	user_8	password_8	Customer
9	9	9	user_9	password_9	Customer

10	10	10	user_10	password_10	Customer
None	None	11	user_11	password_11	Admin
None	None	12	user_12	password_12	Admin
None	None	13	user_13	password_13	Admin
None	None	14	user_14	password_14	Admin
None	None	15	user_15	password_15	Admin
None	None	16	user_16	password_16	Supplier
None	None	17	user_17	password_17	Supplier
None	None	18	user_18	password_18	Supplier
None	None	19	user_19	password_19	Supplier
None	None	20	user_20	password_20	supplier
None	None	21	user_21	password_21	Guider
None	None	22	user_22	password_22	Guider
None	None	23	user_23	password_23	Guider
None	None	24	user_24	password_24	Guider
None	None	25	user_25	password_25	Guider

6. Perform Queries and Sub-queries operations and display the result using SELECT statement

Let's check the order detail from customer

```
%%sql
select c.idOrder,t.idTravelProduct,t.idRoom,t.idFlight,t.platformprofit from CustomerOrder c
inner join TravelProduct t
on c.idTravelProduct = t.idTravelProduct
```

```
 * mysql+pymysql://root:***@localhost/mydb
50 rows affected.
```

idOrder	idTravelProduct	idRoom	idFlight	platformprofit
22	1	30	19	628
25	1	30	19	628
39	1	30	19	628
18	2	22	12	449
31	2	22	12	449
50	2	22	12	449
7	3	21	20	658
19	3	21	20	658
20	3	21	20	658
4	4	7	1	761
35	4	7	1	761
14	5	15	6	520
32	7	28	14	633
36	7	28	14	633
26	8	14	22	416
33	8	14	22	416
40	8	14	22	416
47	8	14	22	416
16	9	15	23	518

24	9	15	23	518
34	9	15	23	518
41	10	13	5	483
42	11	40	20	338
43	11	40	20	338
45	11	40	20	338
38	12	19	23	431
29	13	7	1	432
37	13	7	1	432
2	14	15	2	676
6	14	15	2	676
11	14	15	2	676
12	14	15	2	676
44	14	15	2	676
8	15	21	23	452
17	16	19	19	552
21	16	19	19	552
9	17	7	9	717
10	17	7	9	717
1	18	15	5	307
3	18	15	5	307
5	18	15	5	307
13	18	15	5	307
30	18	15	5	307
49	18	15	5	307
15	19	30	18	662
23	19	30	18	662
28	19	30	18	662
46	19	30	18	662
27	20	41	2	425
48	20	41	2	425

But what if I also want to check the room and filght detail?

Let's use sub-query

```
%%sql
select
o_r.idOrder,o_r.idTravelProduct,o_r.idRoom,o_r.idFlight,o_r.platformprofit,o_r.roomType,o_r.roomDate,f.route,f.flight
Date from (
    select o.idOrder,o.idTravelProduct,o.idRoom,o.idFlight,o.platformprofit,r.roomType,r.roomDate from (
        select c.idOrder,t.idTravelProduct,t.idRoom,t.idFlight,t.platformprofit from CustomerOrder c
        inner join TravelProduct t
        on c.idTravelProduct = t.idTravelProduct) o
    inner join Room r
    on r.idRoom = o.idRoom) o_r
inner join Flight f
on f.idFlight = o_r.idFlight
```

```
 * mysql+pymysql://root:***@localhost/mydb
50 rows affected.
```

idOrder	idTravelProduct	idRoom	idFlight	platformprofit	roomType	roomDate	route	flightDate

22	1	30	19	628	double	2020-05-11	China-France	2020-05-27
25	1	30	19	628	double	2020-05-11	China-France	2020-05-27
39	1	30	19	628	double	2020-05-11	China-France	2020-05-27
18	2	22	12	449	single	2020-06-26	China-France	2020-05-24
31	2	22	12	449	single	2020-06-26	China-France	2020-05-24
50	2	22	12	449	single	2020-06-26	China-France	2020-05-24
7	3	21	20	658	single	2020-05-17	China-Japen	2020-04-23
19	3	21	20	658	single	2020-05-17	China-Japen	2020-04-23
20	3	21	20	658	single	2020-05-17	China-Japen	2020-04-23
4	4	7	1	761	single	2020-05-09	China-America	2020-06-17
35	4	7	1	761	single	2020-05-09	China-America	2020-06-17
14	5	15	6	520	double	2020-05-05	China-America	2020-04-10
32	7	28	14	633	double	2020-04-29	China-America	2020-04-13
36	7	28	14	633	double	2020-04-29	China-America	2020-04-13
26	8	14	22	416	single	2020-05-09	China-France	2020-04-28
33	8	14	22	416	single	2020-05-09	China-France	2020-04-28
40	8	14	22	416	single	2020-05-09	China-France	2020-04-28
47	8	14	22	416	single	2020-05-09	China-France	2020-04-28
16	9	15	23	518	double	2020-05-05	China-Japen	2020-06-22
24	9	15	23	518	double	2020-05-05	China-Japen	2020-06-22
34	9	15	23	518	double	2020-05-05	China-Japen	2020-06-22
41	10	13	5	483	double	2020-06-13	China-America	2020-06-19
42	11	40	20	338	double	2020-04-15	China-Japen	2020-04-23
43	11	40	20	338	double	2020-04-15	China-Japen	2020-04-23
45	11	40	20	338	double	2020-04-15	China-Japen	2020-04-23
38	12	19	23	431	single	2020-06-21	China-Japen	2020-06-22
29	13	7	1	432	single	2020-05-09	China-America	2020-06-17
37	13	7	1	432	single	2020-05-09	China-America	2020-06-17
2	14	15	2	676	double	2020-05-05	China-America	2020-06-10
6	14	15	2	676	double	2020-05-05	China-America	2020-06-10
11	14	15	2	676	double	2020-05-05	China-America	2020-06-10
12	14	15	2	676	double	2020-05-05	China-America	2020-06-10
44	14	15	2	676	double	2020-05-05	China-America	2020-06-10
8	15	21	23	452	single	2020-05-17	China-Japen	2020-06-22
17	16	19	19	552	single	2020-06-21	China-France	2020-05-27
21	16	19	19	552	single	2020-06-21	China-France	2020-05-27
9	17	7	9	717	single	2020-05-09	China-Japen	2020-06-01
10	17	7	9	717	single	2020-05-09	China-Japen	2020-06-01
1	18	15	5	307	double	2020-05-05	China-America	2020-06-19
3	18	15	5	307	double	2020-05-05	China-America	2020-06-19
5	18	15	5	307	double	2020-05-05	China-America	2020-06-19
13	18	15	5	307	double	2020-05-05	China-America	2020-06-19
30	18	15	5	307	double	2020-05-05	China-America	2020-06-19
49	18	15	5	307	double	2020-05-05	China-America	2020-06-19
15	19	30	18	662	double	2020-05-11	China-America	2020-06-17
23	19	30	18	662	double	2020-05-11	China-America	2020-06-17

28	19	30	18	662	double	2020-05-11	China-America	2020-06-17
46	19	30	18	662	double	2020-05-11	China-America	2020-06-17
27	20	41	2	425	single	2020-06-05	China-America	2020-06-10
48	20	41	2	425	single	2020-06-05	China-America	2020-06-10

7. Create four different users (Admin, Tester, Developer and User)

with different privilege settings and display access differentiation among four users. Use GRANT and REVOKE command to give/remove additional access/rights to DB.

7.1 Grant

Statement

```
-- Use ```create``` to create user
create user 'USERNAME'@'HOST' identified by 'PASSWORD';
```

```
-- Use ```grant``` to add new users and authorize them to access MySQL objects
GRANT privilege,[privilege],.. ON privilege_level
TO user [IDENTIFIED BYpassword]
[REQUIREtsl_option]
[WITH[GRANT_OPTION|resource_option]];
```

Grant admin user

```
%%sql
create user 'admin'@'localhost' identified by 'admin';
grant all on mydb.* to 'admin'@'localhost';
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
0 rows affected.




[]
```

Grant customer user

They can only view these tables:

- Order
- TravelProduct

```
%%sql
create user 'customer'@'localhost' identified by 'customer';
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.




[]
```

```
%%sql
grant select on mydb.CustomerOrder to 'customer'@'localhost';
grant select on mydb.TravelProduct to 'customer'@'localhost';
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
0 rows affected.
```

```
[]
```

Grant supplier user

They can only view, update, delete, insert these tables:

- Room
- Flight

```
%%sql
create user 'supplier'@'localhost' identified by 'supplier'
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
```

```
[]
```

```
%%sql
grant select,update,delete,insert on mydb.Room to 'supplier'@'localhost';
grant select,update,delete,insert on mydb.Flight to 'supplier'@'localhost';
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
0 rows affected.
```

```
[]
```

Grant guider user

They can only view table TravelProduct

```
%%sql
create user 'guider'@'localhost' identified by 'guider';
```

```
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
```

```
[]
```

```
%%sql
grant select on mydb.TravelProduct to 'guider'@'localhost';
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.
```

```
[]
```

Let's check if the privilege is work

Take guider as example

```
%sql mysql+pymysql://guider:guider@localhost/mydb
```

```
'Connected: guider@mydb'
```

```
%%sql
select * from TravelProduct
```

```
 * mysql+pymysql://guider:***@localhost/mydb
   mysql+pymysql://root:***@localhost/mydb
20 rows affected.
```

idTravelProduct	idFlight	idRoom	idGuider	description	platformProfit
1	19	30	25	Good	628
2	12	22	21	Mid	449
3	20	21	25	Good	658
4	1	7	25	Mid	761
5	6	15	25	Good	520
6	17	29	25	Perfect	437
7	14	28	25	Mid	633
8	22	14	23	Mid	416
9	23	15	25	Good	518
10	5	13	25	Perfect	483
11	20	40	23	Perfect	338
12	23	19	23	Good	431
13	1	7	22	Mid	432
14	2	15	23	Perfect	676
15	23	21	21	Good	452
16	19	19	24	Mid	552
17	9	7	22	Good	717
18	5	15	25	Mid	307
19	18	30	23	Mid	662
20	2	41	25	Mid	425

```
%%sql
select * from CustomerOrder
```

```
 * mysql+pymysql://guider:***@localhost/mydb
   mysql+pymysql://root:***@localhost/mydb
(pymysql.err.OperationalError) (1142, "SELECT command denied to user 'guider'@'localhost' for table 'customerorder'")
[SQL: select * from CustomerOrder]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

```
%%sql
drop table TravelProduct;
```

```
 * mysql+pymysql://guider:***@localhost/mydb
   mysql+pymysql://root:***@localhost/mydb
(pymysql.err.OperationalError) (1142, "DROP command denied to user 'guider'@'localhost' for table 'travelproduct'")
[SQL: drop table TravelProduct;]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

7.2 Revoke

Statement

```
REVOKE
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    FROM user_or_role [, user_or_role] ...
```

Let's revoke the supplier privilege from table room

```
%sql mysql+pymysql://root:fjwwzx970814@localhost/mydb
```

```
'Connected: root@mydb'
```

```
%%sql
revoke all privileges on mydb.room from 'supplier'@'localhost';
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
0 rows affected.




[]
```

```
%sql mysql+pymysql://supplier:supplier@localhost/mydb
```

```
'Connected: supplier@mydb'
```

```
%%sql
select * from room
```

```
   mysql+pymysql://guider:***@localhost/mydb
   mysql+pymysql://root:***@localhost/mydb
```

```
 * mysql+pymysql://supplier:***@localhost/mydb
(pymysql.err.OperationalError) (1142, "SELECT command denied to user 'supplier'@'localhost' for table 'room'")
[SQL: select * from room]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

The supplier cannot access to table room

```
%%sql
select * from flight
```

```
   mysql+pymysql://guider:***@localhost/mydb
   mysql+pymysql://root:***@localhost/mydb
 * mysql+pymysql://supplier:***@localhost/mydb
25 rows affected.
```

idFlight	idSupplier	route	flightDate	price	ifAvailable
1	16	China-America	2020-06-17	16044	1
2	17	China-America	2020-06-10	11384	1
3	18	China-Japen	2020-05-25	12000	1
4	19	China-Japen	2020-04-19	17422	1
5	20	China-America	2020-06-19	15136	1
6	16	China-America	2020-04-10	10575	1
7	17	China-Japen	2020-05-21	16731	1
8	18	China-Japen	2020-05-13	15579	1
9	19	China-Japen	2020-06-01	19223	1
10	20	China-France	2020-06-10	14325	1
11	16	China-France	2020-04-15	10000	1
12	17	China-France	2020-05-24	12045	1
13	18	China-America	2020-04-11	13643	1
14	19	China-America	2020-04-13	11813	1
15	20	China-Japen	2020-06-27	14727	1
16	16	China-America	2020-04-05	14129	1
17	17	China-France	2020-04-30	13298	1
18	18	China-America	2020-06-17	13064	1
19	19	China-France	2020-05-27	10642	1
20	20	China-Japen	2020-04-23	14763	1
21	16	China-Japen	2020-05-06	18436	1
22	17	China-France	2020-04-28	11112	1
23	18	China-Japen	2020-06-22	17515	1
24	19	China-France	2020-04-14	13450	1
25	20	China-France	2020-04-25	11585	1

8. Create two examples of STORED PROCEDURE and display the results.

Room & Flight Insertion

room

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `insert_room`(
    IN idsupplier int,
    IN roomType varchar(45),
    IN roomDate date,
    IN price decimal(45)
)
BEGIN
```

```
    declare targetSupplier int;

    -- Find the target supplier id
    select s.idSupplier into targetSupplier
    from Supplier s
    where s.idSupplier = idsupplier;

    insert into Room (idSupplier,roomType,roomDate,price,ifAvailable) values
(targetSupplier,roomType,roomDate,price,1);

    select 'succeed';

END
```

```
%%sql
call insert_room(16,"single","2020-06-18",338);
call insert_room(17,"double","2020-06-25",215);
call insert_room(18,"double","2020-04-15",135);
call insert_room(19,"single","2020-06-05",274);
call insert_room(20,"double","2020-06-06",307);
```

```
%%sql
select * from room
```

```
 * mysql+pymysql://root:***@localhost/mydb
21 rows affected.
```

idRoom	idSupplier	roomType	roomDate	price	ifAvailable
7	16	single	2020-05-09	341	1
13	16	double	2020-06-13	281	1
14	17	single	2020-05-09	241	1
15	18	double	2020-05-05	485	1
16	19	single	2020-05-05	382	1
17	20	double	2020-04-16	131	1
18	16	single	2020-04-24	402	1
19	17	single	2020-06-21	144	1
20	18	single	2020-05-25	345	1
21	19	single	2020-05-17	487	1
22	20	single	2020-06-26	151	1
28	16	double	2020-04-29	459	1
29	17	single	2020-05-13	109	1
30	18	double	2020-05-11	288	1
31	19	single	2020-04-03	409	1
32	20	single	2020-05-23	307	1
38	16	single	2020-06-18	338	1
39	17	double	2020-06-25	215	1
40	18	double	2020-04-15	135	1
41	19	single	2020-06-05	274	1
42	20	double	2020-06-06	307	1

Flight

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `insert_flight`(
    IN idsupplier int,
    IN route varchar(45),
    IN flightDate date,
```

```
    IN price decimal(45)
)
BEGIN

    declare targetSupplier int;

    -- Find the target supplier id
    select s.idSupplier into targetSupplier
    from Supplier s
    where s.idSupplier = idsupplier;

    insert into Flight (idSupplier,route,flightDate,price,ifAvailable) values
(targetSupplier,route,flightDate,price,1);

    select 'succeed';
END
```

```
%%sql
call insert_flight(16,"China-Japen","2020-05-06",18436);
call insert_flight(17,"China-France","2020-04-28",11112);
call insert_flight(18,"China-Japen","2020-06-22",17515);
call insert_flight(19,"China-France","2020-04-14",13450);
call insert_flight(20,"China-France","2020-04-25",11585);
```

```
%%sql
select * from Flight
```

```
 * mysql+pymysql://root:***@localhost/mydb
25 rows affected.
```

idFlight	idSupplier	route	flightDate	price	ifAvailable
1	16	China-America	2020-06-17	16044	1
2	17	China-America	2020-06-10	11384	1
3	18	China-Japen	2020-05-25	12000	1
4	19	China-Japen	2020-04-19	17422	1
5	20	China-America	2020-06-19	15136	1
6	16	China-America	2020-04-10	10575	1
7	17	China-Japen	2020-05-21	16731	1
8	18	China-Japen	2020-05-13	15579	1
9	19	China-Japen	2020-06-01	19223	1
10	20	China-France	2020-06-10	14325	1
11	16	China-France	2020-04-15	10000	1
12	17	China-France	2020-05-24	12045	1
13	18	China-America	2020-04-11	13643	1
14	19	China-America	2020-04-13	11813	1
15	20	China-Japen	2020-06-27	14727	1
16	16	China-America	2020-04-05	14129	1
17	17	China-France	2020-04-30	13298	1
18	18	China-America	2020-06-17	13064	1
19	19	China-France	2020-05-27	10642	1
20	20	China-Japen	2020-04-23	14763	1
21	16	China-Japen	2020-05-06	18436	1
22	17	China-France	2020-04-28	11112	1
23	18	China-Japen	2020-06-22	17515	1
24	19	China-France	2020-04-14	13450	1

25	20	China-France	2020-04-25	11585	1

9. Perform COMMIT and ROLLBACK operations with SAVEPOINT/ROLLBACK TO SAVEPOINT.

(Include the use of LOCK/UNLOCK Table operations. Display result of each operation.)

9.1 Transaction

- In MySQL, only databases or tables that use the Innodb database engine support transactions.
- Transaction processing can be used to maintain the integrity of the database, ensuring that batches of SQL statements are all executed or not executed at all.
- Transactions are used to manage insert, update, and delete statements

```
BEGIN -- start a transaction
ROLLBACK -- transaction rollback
COMMIT -- transaction confirmation

SAVEPOINT identifier -- SAVEPOINT allows you to create a savepoint in a transaction, there can be multiple SAVEPOINT
in a transaction;
RELEASE SAVEPOINT identifier -- deletes the savepoint of a transaction. When there is no specified savepoint,
executing this statement will throw an exception;
ROLLBACK TO identifier -- rolls back the transaction to the mark point;
```

lock unlock

```
LOCK TABLE table1 locktype, table2 locktype ...
UNLOCK TABLE[S]
```

```
%sql mysql+pymysql://root:fjwwzx970814@localhost/mydb
```

```
'Connected: root@mydb'
```

Let's try delete some data from a table and rollback

```
27  begin;
28  select * from admin;
29  savepoint point1;
30
31  delete from admin;
32  select * from admin;
33
34  rollback to point1;
35  select * from admin
```

100% 18:29

Result Grid Filter Rows: Search

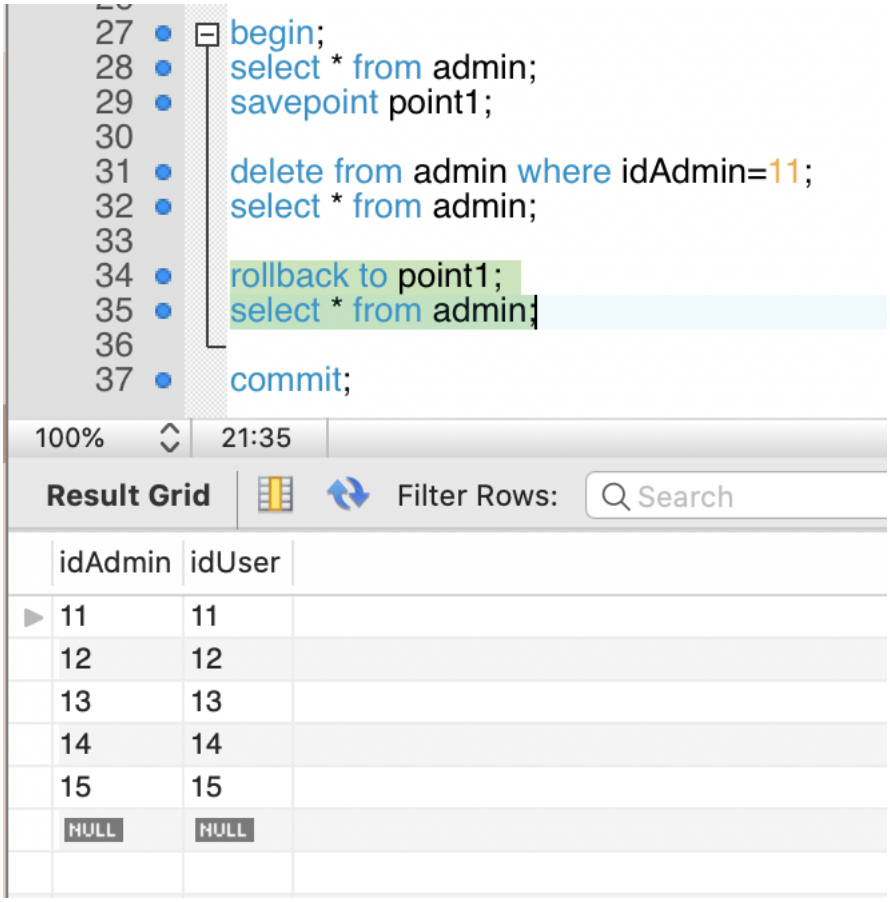
idAdmin	idUser
11	11
12	12
13	13
14	14
15	15
NULL	NULL

```
27  begin;
28  select * from admin;
29  savepoint point1;
30
31  delete from admin where idAdmin=11;
32  select * from admin;
33
34  rollback to point1;
35  select * from admin
```

100% 21:32

Result Grid Filter Rows: Search

idAdmin	idUser
12	12
13	13
14	14
15	15
NULL	NULL



9.2 Lock

There are four types of MySQL LOCK locks, each of which has a different scope:

READ
All users can only read the locked table, and cannot modify the table (including the user who executes LOCK). When the table does not have a WRITE write lock, the READ read lock is executed.

READ LOCAL
The lock performed is the same as READ except that the INSERT command is allowed.

WRITE
Except that the current user is allowed to read and modify the locked table, all access by other users is completely blocked. A WRITE write lock is executed only when all other locks are canceled.

LOW PRIORITY WRITE
Low-priority read locks, during the waiting time (waiting for other locks to be cancelled), other users' access will be considered to have executed the READ read lock, so the waiting time will increase

Note: MySQL lock cannot be used in functions and stored procedures

```
%%sql
show open tables from mydb
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
9 rows affected.
```

Database	Table	In_use	Name_locked
mydb	room	0	0
mydb	customer	0	0
mydb	guider	0	0
mydb	admin	0	0
mydb	supplier	0	0

mydb	user	0	0
mydb	travelproduct	0	0
mydb	customerorder	0	0
mydb	flight	0	0

Let's lock the admin table as read only

```
%%sql
lock tables admin read
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
0 rows affected.
```

```
[]
```

Let's check if it has been locked

```
%%sql
show open tables from mydb
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
9 rows affected.
```

Database	Table	In_use	Name_locked
mydb	room	0	0
mydb	customer	0	0
mydb	guider	0	0
mydb	admin	1	0
mydb	supplier	0	0
mydb	user	0	0
mydb	travelproduct	0	0
mydb	customerorder	0	0
mydb	flight	0	0

```
%%sql
delete from admin where idadmin = 11
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb


---------------------------------------------------------------------------

InternalError                             Traceback (most recent call last)

~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/base.py in _execute_context(self, dialect, constructor,
statement, parameters, *args)
```

```
   1248                     self.dialect.do_execute(
-> 1249                         cursor, statement, parameters, context
   1250                     )


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/default.py in do_execute(self, cursor, statement,
parameters, context)
    579     def do_execute(self, cursor, statement, parameters, context=None):
--> 580         cursor.execute(statement, parameters)
    581


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py in execute(self, query, args)
    169
--> 170         result = self._query(query)
    171         self._executed = query


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py in _query(self, q)
    327         self._clear_result()
--> 328         conn.query(q)
    329         self._do_get_result()


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in query(self, sql, unbuffered)
    516         self._execute_command(COMMAND.COM_QUERY, sql)
--> 517         self._affected_rows = self._read_query_result(unbuffered=unbuffered)
    518         return self._affected_rows


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in _read_query_result(self, unbuffered)
    731             result = MySQLResult(self)
--> 732             result.read()
    733         self._result = result


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in read(self)
   1074         try:
-> 1075             first_packet = self.connection._read_packet()
   1076


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in _read_packet(self, packet_type)
    683         packet = packet_type(buff, self.encoding)
--> 684         packet.check_error()
    685         return packet


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/protocol.py in check_error(self)
    219             if DEBUG: print("errno =", errno)
--> 220             err.raise_mysql_exception(self._data)
    221


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/err.py in raise_mysql_exception(data)
    108     errorclass = error_map.get(errno, InternalError)
--> 109     raise errorclass(errno, errval)


InternalError: (1099, "Table 'admin' was locked with a READ lock and can't be updated")


The above exception was the direct cause of the following exception:


InternalError                             Traceback (most recent call last)

<ipython-input-70-5b1b0d9659a2> in <module>
----> 1 get_ipython().run_cell_magic('sql', '', 'delete from admin where idadmin = 11\n')


~/opt/anaconda3/lib/python3.7/site-packages/IPython/core/interactiveshell.py in run_cell_magic(self, magic_name,
line, cell)
   2357             with self.builtin_trap:
   2358                 args = (magic_arg_s, cell)
-> 2359                 result = fn(*args, **kwargs)
   2360             return result
```

```
   2361 


</Users/brickeawang/opt/anaconda3/lib/python3.7/site-packages/decorator.py:decorator-gen-157> in execute(self, line,
cell, local_ns)


~/opt/anaconda3/lib/python3.7/site-packages/IPython/core/magic.py in <lambda>(f, *a, **k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188 
    189         if callable(arg):


</Users/brickeawang/opt/anaconda3/lib/python3.7/site-packages/decorator.py:decorator-gen-156> in execute(self, line,
cell, local_ns)


~/opt/anaconda3/lib/python3.7/site-packages/IPython/core/magic.py in <lambda>(f, *a, **k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188 
    189         if callable(arg):


~/opt/anaconda3/lib/python3.7/site-packages/sql/magic.py in execute(self, line, cell, local_ns)
     93 
     94         try:
---> 95             result = sql.run.run(conn, parsed['sql'], self, user_ns)
     96 
     97             if result is not None and not isinstance(result, str) and self.column_local_vars:


~/opt/anaconda3/lib/python3.7/site-packages/sql/run.py in run(conn, sql, config, user_namespace)
    338             else:
    339                 txt = sqlalchemy.sql.text(statement)
--> 340                 result = conn.session.execute(txt, user_namespace)
    341             _commit(conn=conn, config=config)
    342             if result and config.feedback:


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/base.py in execute(self, object_, *multiparams,
**params)
    986             raise exc.ObjectNotExecutableError(object_)
    987         else:
--> 988             return meth(self, multiparams, params)
    989 
    990     def _execute_function(self, func, multiparams, params):


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/sql/elements.py in _execute_on_connection(self, connection,
multiparams, params)
    285     def _execute_on_connection(self, connection, multiparams, params):
    286         if self.supports_execution:
--> 287             return connection._execute_clauseelement(self, multiparams, params)
    288         else:
    289             raise exc.ObjectNotExecutableError(self)


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/base.py in _execute_clauseelement(self, elem,
multiparams, params)
   1105             distilled_params,
   1106             compiled_sql,
-> 1107             distilled_params,
   1108         )
   1109         if self._has_events or self.engine._has_events:


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/base.py in _execute_context(self, dialect, constructor,
statement, parameters, *args)
   1251         except BaseException as e:
   1252             self._handle_dbapi_exception(
-> 1253                 e, statement, parameters, cursor, context
   1254             )
   1255 
```

```
~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/base.py in _handle_dbapi_exception(self, e, statement,
parameters, cursor, context)
   1471                 util.raise_from_cause(newraise, exc_info)
   1472             elif should_wrap:
-> 1473                 util.raise_from_cause(sqlalchemy_exception, exc_info)
   1474             else:
   1475                 util.reraise(*exc_info)


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/util/compat.py in raise_from_cause(exception, exc_info)
    396     exc_type, exc_value, exc_tb = exc_info
    397     cause = exc_value if exc_value is not exception else None
--> 398     reraise(type(exception), exception, tb=exc_tb, cause=cause)
    399
    400


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/util/compat.py in reraise(tp, value, tb, cause)
    150             value.__cause__ = cause
    151         if value.__traceback__ is not tb:
--> 152             raise value.with_traceback(tb)
    153         raise value
    154


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/base.py in _execute_context(self, dialect, constructor,
statement, parameters, *args)
   1247                 if not evt_handled:
   1248                     self.dialect.do_execute(
-> 1249                         cursor, statement, parameters, context
   1250                     )
   1251         except BaseException as e:


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/default.py in do_execute(self, cursor, statement,
parameters, context)
    578
    579     def do_execute(self, cursor, statement, parameters, context=None):
--> 580         cursor.execute(statement, parameters)
    581
    582     def do_execute_no_params(self, cursor, statement, context=None):


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py in execute(self, query, args)
    168         query = self.mogrify(query, args)
    169
--> 170         result = self._query(query)
    171         self._executed = query
    172         return result


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py in _query(self, q)
    326         self._last_executed = q
    327         self._clear_result()
--> 328         conn.query(q)
    329         self._do_get_result()
    330         return self.rowcount


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in query(self, sql, unbuffered)
    515                 sql = sql.encode(self.encoding, 'surrogateescape')
    516         self._execute_command(COMMAND.COM_QUERY, sql)
--> 517         self._affected_rows = self._read_query_result(unbuffered=unbuffered)
    518         return self._affected_rows
    519


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in _read_query_result(self, unbuffered)
    730         else:
    731             result = MySQLResult(self)
--> 732             result.read()
    733         self._result = result
    734         if result.server_status is not None:
```

```
~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in read(self)
   1073     def read(self):
   1074         try:
-> 1075             first_packet = self.connection._read_packet()
   1076
   1077             if first_packet.is_ok_packet():


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in _read_packet(self, packet_type)
    682
    683         packet = packet_type(buff, self.encoding)
--> 684         packet.check_error()
    685         return packet
    686


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/protocol.py in check_error(self)
    218             errno = self.read_uint16()
    219             if DEBUG: print("errno =", errno)
--> 220             err.raise_mysql_exception(self._data)
    221
    222     def dump(self):


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/err.py in raise_mysql_exception(data)
    107         errval = data[3:].decode('utf-8', 'replace')
    108     errorclass = error_map.get(errno, InternalError)
--> 109     raise errorclass(errno, errval)


InternalError: (pymysql.err.InternalError) (1099, "Table 'admin' was locked with a READ lock and can't be updated")
[SQL: delete from admin where idadmin = 11]
(Background on this error at: http://sqlalche.me/e/2j85)
```

As error message shown above, the admin table has been locked

```
%%sql
unlock tables
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
0 rows affected.




[]
```

```
%%sql
show open tables from mydb
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
9 rows affected.
```

Database	Table	In_use	Name_locked
mydb	room	0	0
mydb	customer	0	0
mydb	guider	0	0
mydb	admin	0	0
mydb	supplier	0	0

mydb	user	0	0
mydb	travelproduct	0	0
mydb	customerorder	0	0
mydb	flight	0	0

10. Perform Triggers operations using INSERT, UPDATE, DELETE and display result

I have create insert update delete trigger in admin table

```
%%sql
delete from admin where idAdmin =11
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
1 rows affected.




[]
```

```
%%sql
select * from AdminLog
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
1 rows affected.
```

idAdminLog	idAdmin	idUser	message	happendDate
1	11	11	delete	2020-04-16

```
%%sql
insert into Admin (idAdmin,idUser) values(11,11)
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
1 rows affected.




[]
```

```
%%sql
select * from AdminLog
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
2 rows affected.
```

idAdminLog	idAdmin	idUser	message	happendDate

1	11	11	delete	2020-04-16
2	11	11	insert	2020-04-16

```
%%sql
update Admin set idAdmin = 100 where idAdmin = 11
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
1 rows affected.
```

```
[]
```

```
%%sql
select * from AdminLog
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
3 rows affected.
```

idAdminLog	idAdmin	idUser	message	happendDate
1	11	11	delete	2020-04-16
2	11	11	insert	2020-04-16
3	100	11	update	2020-04-16

11. Perform three examples of VIEW operation simple, and with increasing Complexities

Check admin with user inforamtion

```
CREATE VIEW `check_admin` AS
select a.idadmin,a.iduser,u.username from admin a
inner join user u
on a.iduser = u.iduser
```

Check customer order detail

```
USE `mydb`;
CREATE  OR REPLACE VIEW `check_customer_order` AS
select c.idOrder,t.idTravelProduct,t.idRoom,t.idFlight,t.platformprofit from CustomerOrder c
inner join TravelProduct t
on c.idTravelProduct = t.idTravelProduct;;
```

Check travel product detail with room and flight information

```
USE `mydb`;
CREATE  OR REPLACE VIEW `check_travel_product_detail` AS
select
o_r.idOrder,o_r.idTravelProduct,o_r.idRoom,o_r.idFlight,o_r.platformprofit,o_r.roomType,o_r.roomDate,f.route,f.flight
Date from (
    select o.idOrder,o.idTravelProduct,o.idRoom,o.idFlight,o.platformprofit,r.roomType,r.roomDate from (
        select c.idOrder,t.idTravelProduct,t.idRoom,t.idFlight,t.platformprofit from CustomerOrder c
        inner join TravelProduct t
        on c.idTravelProduct = t.idTravelProduct) o
    inner join Room r
    on r.idRoom = o.idRoom) o_r
```

```
inner join Flight f
on f.idFlight = o_r.idFlight;
```

```
%%sql
select * from check_admin
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
5 rows affected.
```

idadmin	iduser	username
100	11	user_11
12	12	user_12
13	13	user_13
14	14	user_14
15	15	user_15

```
%%sql
select * from check_customer_order
```

```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
50 rows affected.
```

idOrder	idTravelProduct	idRoom	idFlight	platformprofit
22	1	30	19	628
25	1	30	19	628
39	1	30	19	628
18	2	22	12	449
31	2	22	12	449
50	2	22	12	449
7	3	21	20	658
19	3	21	20	658
20	3	21	20	658
4	4	7	1	761
35	4	7	1	761
14	5	15	6	520
32	7	28	14	633
36	7	28	14	633
26	8	14	22	416
33	8	14	22	416
40	8	14	22	416
47	8	14	22	416
16	9	15	23	518
24	9	15	23	518
34	9	15	23	518
41	10	13	5	483

42	11	40	20	338
43	11	40	20	338
45	11	40	20	338
38	12	19	23	431
29	13	7	1	432
37	13	7	1	432
2	14	15	2	676
6	14	15	2	676
11	14	15	2	676
12	14	15	2	676
44	14	15	2	676
8	15	21	23	452
17	16	19	19	552
21	16	19	19	552
9	17	7	9	717
10	17	7	9	717
1	18	15	5	307
3	18	15	5	307
5	18	15	5	307
13	18	15	5	307
30	18	15	5	307
49	18	15	5	307
15	19	30	18	662
23	19	30	18	662
28	19	30	18	662
46	19	30	18	662
27	20	41	2	425
48	20	41	2	425

```
%%sql
select * from check_travel_product_detail
```

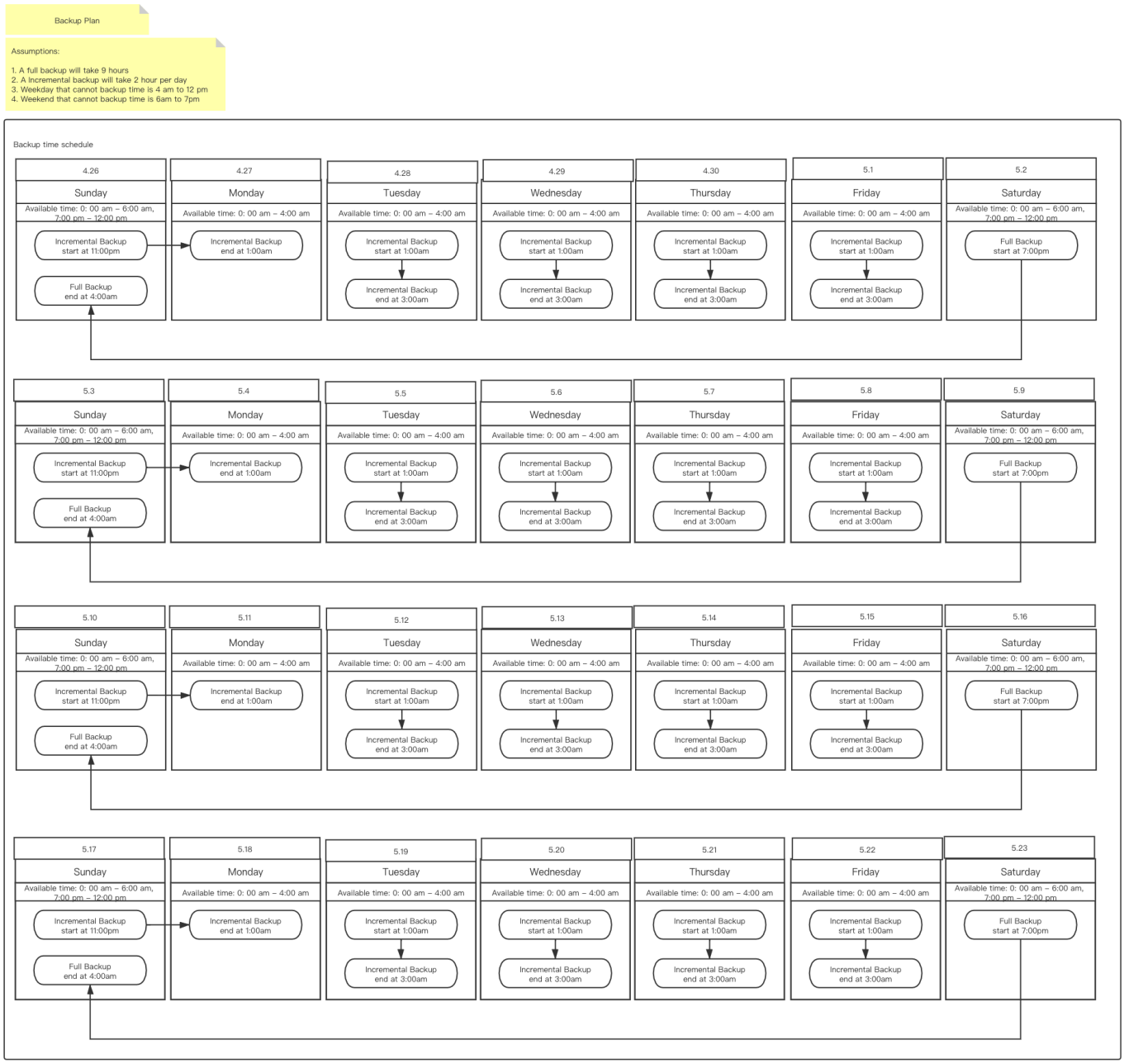
```
   mysql+pymysql://guider:***@localhost/mydb
 * mysql+pymysql://root:***@localhost/mydb
   mysql+pymysql://supplier:***@localhost/mydb
50 rows affected.
```

idOrder	idTravelProduct	idRoom	idFlight	platformprofit	roomType	roomDate	route	flightDate
22	1	30	19	628	double	2020-05-11	China-France	2020-05-27
25	1	30	19	628	double	2020-05-11	China-France	2020-05-27
39	1	30	19	628	double	2020-05-11	China-France	2020-05-27
18	2	22	12	449	single	2020-06-26	China-France	2020-05-24
31	2	22	12	449	single	2020-06-26	China-France	2020-05-24
50	2	22	12	449	single	2020-06-26	China-France	2020-05-24
7	3	21	20	658	single	2020-05-17	China-Japen	2020-04-23
19	3	21	20	658	single	2020-05-17	China-Japen	2020-04-23
20	3	21	20	658	single	2020-05-17	China-Japen	2020-04-23
4	4	7	1	761	single	2020-05-09	China-America	2020-06-17

35	4	7	1	761	single	2020-05-09	China-America	2020-06-17
14	5	15	6	520	double	2020-05-05	China-America	2020-04-10
32	7	28	14	633	double	2020-04-29	China-America	2020-04-13
36	7	28	14	633	double	2020-04-29	China-America	2020-04-13
26	8	14	22	416	single	2020-05-09	China-France	2020-04-28
33	8	14	22	416	single	2020-05-09	China-France	2020-04-28
40	8	14	22	416	single	2020-05-09	China-France	2020-04-28
47	8	14	22	416	single	2020-05-09	China-France	2020-04-28
16	9	15	23	518	double	2020-05-05	China-Japen	2020-06-22
24	9	15	23	518	double	2020-05-05	China-Japen	2020-06-22
34	9	15	23	518	double	2020-05-05	China-Japen	2020-06-22
41	10	13	5	483	double	2020-06-13	China-America	2020-06-19
42	11	40	20	338	double	2020-04-15	China-Japen	2020-04-23
43	11	40	20	338	double	2020-04-15	China-Japen	2020-04-23
45	11	40	20	338	double	2020-04-15	China-Japen	2020-04-23
38	12	19	23	431	single	2020-06-21	China-Japen	2020-06-22
29	13	7	1	432	single	2020-05-09	China-America	2020-06-17
37	13	7	1	432	single	2020-05-09	China-America	2020-06-17
2	14	15	2	676	double	2020-05-05	China-America	2020-06-10
6	14	15	2	676	double	2020-05-05	China-America	2020-06-10
11	14	15	2	676	double	2020-05-05	China-America	2020-06-10
12	14	15	2	676	double	2020-05-05	China-America	2020-06-10
44	14	15	2	676	double	2020-05-05	China-America	2020-06-10
8	15	21	23	452	single	2020-05-17	China-Japen	2020-06-22
17	16	19	19	552	single	2020-06-21	China-France	2020-05-27
21	16	19	19	552	single	2020-06-21	China-France	2020-05-27
9	17	7	9	717	single	2020-05-09	China-Japen	2020-06-01
10	17	7	9	717	single	2020-05-09	China-Japen	2020-06-01
1	18	15	5	307	double	2020-05-05	China-America	2020-06-19
3	18	15	5	307	double	2020-05-05	China-America	2020-06-19
5	18	15	5	307	double	2020-05-05	China-America	2020-06-19
13	18	15	5	307	double	2020-05-05	China-America	2020-06-19
30	18	15	5	307	double	2020-05-05	China-America	2020-06-19
49	18	15	5	307	double	2020-05-05	China-America	2020-06-19
15	19	30	18	662	double	2020-05-11	China-America	2020-06-17
23	19	30	18	662	double	2020-05-11	China-America	2020-06-17
28	19	30	18	662	double	2020-05-11	China-America	2020-06-17
46	19	30	18	662	double	2020-05-11	China-America	2020-06-17
27	20	41	2	425	single	2020-06-05	China-America	2020-06-10
48	20	41	2	425	single	2020-06-05	China-America	2020-06-10

12. Backup plan

Create a backup plan of your choice, state your assumptions/constraints and explain your reasoning. Print your selection on a sample monthly calendar (say 30 day) with FB for Full Backup and IB for Incremental Backup



10 True/False Question

1. CHAR column length is fixed to the length declared when creating the table VARCHAR column is of indefinite length, but it cannot exceed the declared length

- True

2. The primary key is also a candidate key. By convention, candidate keys can be designated as primary keys and can be used for any foreign key reference.

- True

3. Function `FORMAT (X, D)` -format number D in X format

- False
  Right:Function `FORMAT (X, D)` -format number X to D significant digit

4. When using `order by`. The result, by default, it is sorted in ascending order.

- True

5. Use the percent character `%` in the LIKE clause to represent any character

- True

6. When using `DELETE FROM table_name [WHERE Clause]`, if no WHERE clause is specified, the particular table in the MySQL will be deleted.

- False
  Right: When using `DELETE FROM table_name [WHERE Clause]`, if no WHERE clause is specified, all records in the MySQL table will be deleted.

7. The TRUNCATE command permanently deletes one particular row from the table

- False
  Right: The TRUNCATE command permanently deletes each row from the table

8. `BEGIN` or `START` explicitly start a transaction;

- False
  Right: `BEGIN` or `START TRANSACTION` explicitly start a transaction;

9. By default, when you disconnect from the database, the temporary table will be automatically destroyed

- True

10. Find if the table column in the data table is `NULL`, you can use `IS NULL` and `IS NOT NULL` or `column = NULL` and `column <> NULL`

- False
  Right: Find if the runoob_test_tbl column in the data table is NULL, you must use IS NULL and IS NOT NULL

10 Multiple choice questions

1. The following is the trigger in the Mysql table

```
A. BEFORE INSERT
B. AFTER INSERT
C. AFTER CREATE
D. BEFORE UPDATE
```

Answer: A, B, D

2. The "show databases like' student% '" command can display the following databases ()

```
A. student_my
B. studenty
C. mystudent
D. student
```

Answer: A, B, D

3. SQL language sets several functional modules into one, including ()

```
A. DCL
B. DML
C. DNL
D. DDL
```

Answer: A, B, D

4. The following statement is correct ()

```
A. alter table user drop column sex;
B. alter table user add sex varchar (20);
C. alter table user drop sex;
D. alter table user modify id int primary key;
```

Answer: A, B, C, D

5. Views are generally not used for which of the following statements ()

```
A. DELETE
B. SELECT
C. INSERT
D. UPDATE
```

Answer: A, C, D

6. Regarding the sorting of search results, the correct one is ()

```
A. The keyword DESC means descending order, ASC means ascending order
B. If you specify a multi-column sort, you can only use the ascending or descending keywords in the last column
C. If you specify multi-column sorting, you can use ascending or descending keywords in any column
D. The keyword ASC means descending, DESC means ascending
```

Answer: A, C

7. The statement select * from products where prod_name like '% se%' result set includes ()

```
A. Retrieve the data in the products table whose prod_name field ends with 'se'
B. Retrieve the data with the 'se' switch in the prod_name field in the products table
C. Retrieve the data containing 'se' in the prod_name field in the products table
D. Retrieve data where the prod_name field in the products table does not contain 'se'
```

Answer: A, B, C

8. In regular expressions, repeated metacharacters "*" means ()

```
A. No match
B. Match only 1
C. 0 matches
D. Multiple matches
```

Answer: C, D

9. The following description of union is correct ()

```
A. union only connects query statements with the exact same result set
B. union can connect multiple result sets with the same number of data types in the result set
C. union is the key word for filtering, and then operate on the result set
D. Any query can be connected using union
```

Answer: A, C, D

10. What logical operators does mySQL support

```
A. &&
B. ||
C. NOT
D. AND
```

Answer: C, D

3 Normalization questions

1. Please do 1NF normalization for following table

StudentID	Name	Department	Course
401	Akon	CSE	c1, c2
402	Bkon	CSE	c1, c2
403	Ckon	CSE	c1
404	Dkon	CSE	c2

Answer:

StudentID	Name	Department	Course
401	Akon	CSE	c1
402	Bkon	CSE	c1
401	Akon	CSE	c2
402	Bkon	CSE	c2
403	Ckon	CSE	c1
404	Dkon	CSE	c2

2. Please find partial functional dependency in this table

StudentID	Name	CourseID	Score	Teacher
001	A	L1	90	Mr.X
001	A	L2	80	Mr.Y
002	B	L2	910	Mr.Y

Answer:

[Student ID + Course ID] You can determine any score or student ID or teacher together, so [Student ID + Course ID] is the candidate key for this form. A table can have multiple keys (key), generally we will choose one of them as the primary key.
At this time, we can see that the attribute of the teacher is actually only determined by the course number, and the course number is only a part of the candidate key, so at this time we say that the teacher attribute has some functional dependencies.

3. Please do make the table meet the 3NF

StudentID	Name	CourseID	Score	Teacher
001	A	L1	90	Mr.X
001	A	L2	80	Mr.Y
002	B	L2	910	Mr.Y

Answer:

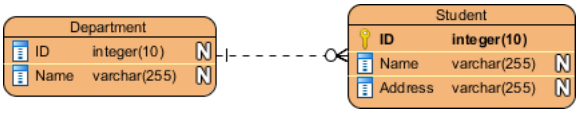
StudentID	Name
001	A
002	B

CourseID	Teacher
L1	Mr. X
L2	Mr. Y
L3	Mr. Y

Teacher	Department
Mr. X	CS
Mr. Y	EE

2 ERD and EERD concepts questions

1. What is the relationship between these two enities?



Answer: One Department has zero/one/many Student

2. What are the general constraints of ERD diagrams?

Answer:

- One to one
- One to many
- Many to many

5 SQL coding skills and concept questions

1. Please write the code to return the visit history of all websites

```
website table
+----+--------------+---------------------------+-------+---------+
| id | name         | url                       | alexa | country |
+----+--------------+---------------------------+-------+---------+
| 1  | Google       | https://www.google.cm/    | 1     | USA     |
| 2  | taobao       | https://www.taobao.com/   | 13    | CN      |
| 3  | ronoob       | http://www.runoob.com/    | 4689  | CN      |
| 4  | weibo        | http://weibo.com/         | 20    | CN      |
| 5  | Facebook     | https://www.facebook.com/ | 3     | USA     |
| 7  | stackoverflow| http://stackoverflow.com/ |   0   | IND     |
```

```
+----+---------------+--------------------------+-------+---------+
access_log table
+-----+---------+-------+------------+
| aid | site_id | count | date       |
+-----+---------+-------+------------+
|   1 |       1 |    45 | 2016-05-10 |
|   2 |       3 |   100 | 2016-05-13 |
|   3 |       1 |   230 | 2016-05-14 |
|   4 |       2 |    10 | 2016-05-14 |
|   5 |       5 |   205 | 2016-05-14 |
|   6 |       4 |    13 | 2016-05-15 |
|   7 |       3 |   220 | 2016-05-15 |
|   8 |       5 |   545 | 2016-05-16 |
|   9 |       3 |   201 | 2016-05-17 |
+-----+---------+-------+------------+
```

Answer:

```
SELECT websites.name, access_log.count, access_log.date
FROM websites
INNER JOIN access_log
ON Websites.id=access_log.site_id
ORDER BY access_log.count;
```

2. Use sql to define a table with column (ID not null, lastname not null, firstname, age can be null)

Answer:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
```

3. Copy all columns from one table and insert into another existing table

Answer:

```
INSERT INTO table2
SELECT * FROM table1;
```

4. Use SQL statement selects all customers whose name starts with the letter "G"

```
website table
+----+--------------+---------------------------+-------+---------+
| id | name         | url                       | alexa | country |
+----+--------------+---------------------------+-------+---------+
| 1  | Google       | https://www.google.cm/    | 1     | USA     |
| 2  | taobao       | https://www.taobao.com/   | 13    | CN      |
| 3  | ronoob       | http://www.runoob.com/    | 4689  | CN      |
| 4  | weibo        | http://weibo.com/         | 20    | CN      |
| 5  | Facebook     | https://www.facebook.com/ | 3     | USA     |
| 7  | stackoverflow| http://stackoverflow.com/ |   0   | IND     |
+----+--------------+---------------------------+-------+---------+
```

Answer:

```
SELECT * FROM Websites
WHERE name LIKE 'G%';
```

5. Find websites with total visits greater than 200

```
website table
+----+--------------+---------------------------+-------+---------+
| id | name         | url                       | alexa | country |
+----+--------------+---------------------------+-------+---------+
| 1  | Google       | https://www.google.cm/    | 1     | USA     |
```

```
| 2  | taobao       | https://www.taobao.com/   | 13    | CN      |
| 3  | ronoob       | http://www.runoob.com/    | 4689  | CN      |
| 4  | weibo        | http://weibo.com/         | 20    | CN      |
| 5  | Facebook     | https://www.facebook.com/ | 3     | USA     |
| 7  | stackoverflow| http://stackoverflow.com/ |   0   | IND     |
+----+--------------+---------------------------+-------+---------+
access_log table
+-----+---------+-------+------------+
| aid | site_id | count | date       |
+-----+---------+-------+------------+
|   1 |       1 |    45 | 2016-05-10 |
|   2 |       3 |   100 | 2016-05-13 |
|   3 |       1 |   230 | 2016-05-14 |
|   4 |       2 |    10 | 2016-05-14 |
|   5 |       5 |   205 | 2016-05-14 |
|   6 |       4 |    13 | 2016-05-15 |
|   7 |       3 |   220 | 2016-05-15 |
|   8 |       5 |   545 | 2016-05-16 |
|   9 |       3 |   201 | 2016-05-17 |
+-----+---------+-------+------------+
```

Answer:

```
SELECT Websites.name, Websites.url, SUM(access_log.count) AS nums FROM (access_log
INNER JOIN Websites
ON access_log.site_id=Websites.id)
GROUP BY Websites.name
HAVING SUM(access_log.count) > 200;
```

Team Evaluation

Write a team evaluation (for each of your team members—2-3 sentences for each team member) including who do you think is the most valuable team member and why; Who did what according to your own observations? Rank everyone on your team including yourself (1 is the best, 2 is 2nd best, etc.) and why they are ranked in that order (no equal contribution will be an acceptable answer); what works best for your team and what did not? Rank all the team in class including your team.

Zixiao(most valuable team member)

Rank: 1

I choose myself as the most valuable team member is because I think I almost participated in everything in this project.

Maybe evaluating myself like this will make others think I am arrogant.

But to be honest, I make two important breakthroughs (RoadMap, Multi-floor database design) on my own.

I implement most of the code for the project, which includes map data import, positioning algorithm implementation, pathfinding algorithm implementation, and single-floor navigation simulation. And I familiar with every part in this project.

I drive a lot of my passion for this project and try to do as much as possible. And I will make this project to be a deliverable application if I have resource and time.

So this is why I choose myself as the most valuable team member.

Nityashree

Rank: 2

Evaluation:

```
Very motivated and very willing to learn knowledge in new fields. She has good skills in Excel and database design
```

Work during project:

```
* Complete trigger implementation, some function implementation, some store procedure implementation and database
user privilege management
* Complete database backup plan
* Contribute to ERD diagram design
* Contribute to database implementation
* Contribute to use case, sequence diagram design
* Contribute to project proposal, final presentation and final report
* Contribute to project research
```

Jalal

Rank: 3

Evaluation:

```
Humor and optimism, not afraid of areas that are not familiar at. He has no experience in SQL coding before. But he
still work hard to learn it.
```

Work during project:

```
* Contribute to position technology research
* Contribute to position algorithm research
* Contribute to AutoCAD blueprint implementation and excel transfermation
* Contribute to ERD diagram design
* Contribute to database implementation
* Contribute to use case, sequence diagram design
* Contribute to project proposal, final presentation and final report
* Contribute to project research
```

Saurin

Rank: 4

Evaluation:

```
Optimistic and willing to contribute. He has many whimsical ideas.
```

Work during project:

```
* First one to propose the 'indoor navigation' idea. (Actually I didn't quite remember who is the first one came out
with this idea. But in my memary, he was)
* Contribute to AutoCAD blueprint implementation and excel transfermation
* Contribute to ERD diagram design
* Contribute to database implementation
* Contribute to use case, sequence diagram design
* Contribute to project proposal, final presentation and final report
* Contribute to project research
```

Vaishnav

Rank: 5

Evaluation:

```
Vaishnav has ability to do a better job. But he did not invest enough time in the team project.
```

Work during project:

```
* Complete UI design
* Complete UI xd file
```

TA Opportunity

If you are given an opportunity to be a paid TA or non-paid/volunteer in Fall 2020/Spring 2021, would you be interested (Do include if you are interested in volunteer vs paid position?) If you are interested, what skillsets and quality would make you a successful TA or a part of the team?

I'm interested in being a paid TA.

Skillsets:

- SQL
- MySQL
- WorkBench
- Database Design
- Database Management
- Program Design

- Python
- Java

Quality:

- Teaching experience (Used to be volunteer teach for supported education)
- Good English communication skill
- Patiently
- Time management
- Responsible