CW2 Implementation of Database

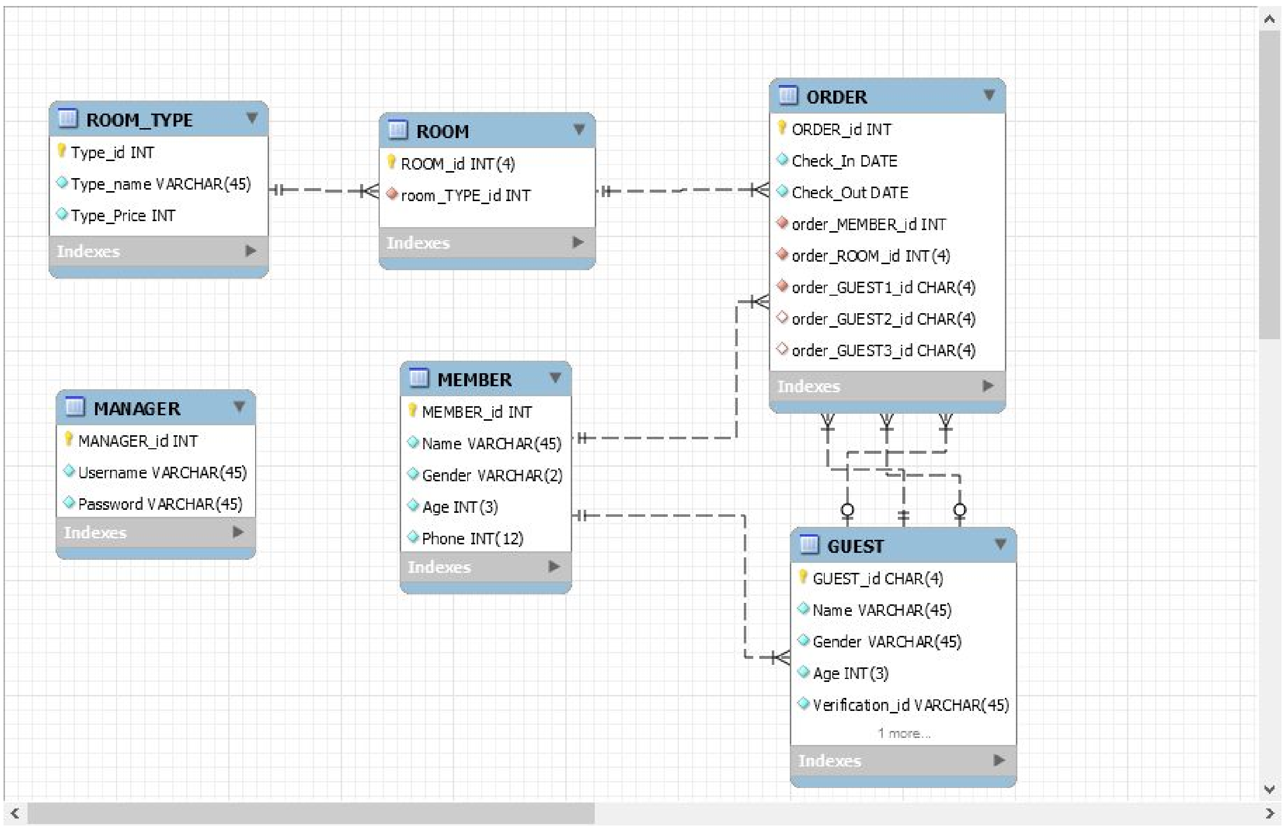
——Hotel Booking System

by Jiaxue Tian, Dian Ding, Group 11

1. relational schema from CW1

Some changes have been made based on the feedback from Tony.

Table STATUS and PRICE have been removed because the two attributes can be calculated. Attribute Verification_id has been added into TABLE ORDERS in order to track the check-in records of specific guests. Verification_id includes passport number, driving license number, national id number and others, it can identify the guests in case there would be multiple guests with the same name .



- Room(id, type); PK:id; FK:id, type;
- RoomType(type, Type_name, price); PK and FK: type;
- Table MEMBER (MEMBER_id, Name, Gender, Age, Phone); PK and FK: id.
- Table GUEST (GUEST_id, Name, Gender, Age, Verification_id, guest_member_id); PK: GUEST_id; FK: guest_member_id;
- Table ORDERS (order_id,check_in, check_out, order_member_id, order_room_id, order_GUEST1_id, order_GUEST2_id, order_GUEST3_id,). PK:order_id, FK:order_id, order_member_id, order_room_id.
- Table MANAGER (manager_id, username, Password); PK and FK: manager_id.

2. CREATE TABLE Commands

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

-- Table ROOM_TYPE

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

DROP TABLE ROOM_TYPE ;

CREATE TABLE ROOM_TYPE (

  Type_id INT NOT NULL,

  Type_name VARCHAR(45) NOT NULL,

  Type_Price INT NOT NULL,

  PRIMARY KEY (Type_id))

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

-- Table ROOM

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

DROP TABLE ROOM ;
```

```
CREATE TABLE ROOM (

  ROOM_id NUMBER(4) NOT NULL,

  room_TYPE_id INT NOT NULL,

  PRIMARY KEY (ROOM_id),

  CONSTRAINT fk_ROOM_TYPE

    FOREIGN KEY (room_TYPE_id)

    REFERENCES ROOM_TYPE (Type_id))

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

-- Table MEMBER

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

DROP TABLE MEMBER ;

CREATE TABLE MEMBER (

  MEMBER_id INT NOT NULL,

  Name VARCHAR(45) NOT NULL,

  Gender VARCHAR(2) NOT NULL,

  Age NUMBER(3) NOT NULL,

  Phone INT NOT NULL,

  PRIMARY KEY (MEMBER_id))

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

-- Table GUEST

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

DROP TABLE GUEST ;

CREATE TABLE GUEST (
  GUEST_id CHAR(4) NOT NULL,

  Name VARCHAR(45) NOT NULL,

  Gender VARCHAR(45) NOT NULL,

  Age NUMBER(3) NOT NULL,

  Verification_id VARCHAR(45) NOT NULL,

  GUEST_MEMBER_id INT NOT NULL,

  PRIMARY KEY (GUEST_id),

  CONSTRAINT fk_MEMBER_GUEST

    FOREIGN KEY (GUEST_MEMBER_id)

    REFERENCES MEMBER (MEMBER_id))

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

-- Table ORDER

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

DROP TABLE ORDERS ;

CREATE TABLE ORDERS (

  ORDER_id INT NOT NULL,

  Check_In DATE NOT NULL,

  Check_Out DATE NOT NULL,

  order_MEMBER_id INT NOT NULL,

  order_ROOM_id NUMBER(4) NOT NULL,

  order_GUEST1_id CHAR(4) NOT NULL,

  order_GUEST2_id CHAR(4) NULL,

  order_GUEST3_id CHAR(4) NULL,
 PRIMARY KEY (ORDER_id))

alter table orders

add CONSTRAINT fk_ORDER_MEMBER

  FOREIGN KEY (order_MEMBER_id)

  REFERENCES MEMBER(MEMBER_id)

alter table orders

add CONSTRAINT fk_ORDER_ROOM

  FOREIGN KEY (order_ROOM_id)

  REFERENCES ROOM(ROOM_id)

alter table orders

add CONSTRAINT fk_ORDER_GUEST1

  FOREIGN KEY (order_GUEST1_id)

  REFERENCES GUEST (GUEST_id)

alter table orders

add CONSTRAINT fk_ORDER_GUEST2

  FOREIGN KEY (order_GUEST2_id)

  REFERENCES GUEST (GUEST_id)

alter table orders

add CONSTRAINT fk_ORDER_GUEST3

  FOREIGN KEY (order_GUEST3_id)

  REFERENCES GUEST (GUEST_id)

-- Multiple ALTER commands were used above for the reason that if we insert all the constraints
together with the CREATE commands, errors were always occurred.
```

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

-- Table MANAGER

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

DROP TABLE MANAGER ;

CREATE TABLE MANAGER (

  MANAGER_id INT NOT NULL,

  Username VARCHAR(45) NOT NULL,

  Password VARCHAR(45) NOT NULL,

 PRIMARY KEY (MANAGER_id))
```

3. Sample Test Data

room_type:

```
INSERT INTO room_type VALUES ('1', 'standard', '100');

INSERT INTO room_type VALUES ('2', 'king', '200');

INSERT INTO room_type VALUES ('3', 'luxury', '400');
```

room:

```
INSERT INTO room VALUES ('101', '1');

INSERT INTO room VALUES ('102', '1');

INSERT INTO room VALUES ('103', '1');

INSERT INTO room VALUES ('104', '1');

INSERT INTO room VALUES ('105', '1');
INSERT INTO room VALUES ('201', '2');

INSERT INTO room VALUES ('202', '2');

INSERT INTO room VALUES ('203', '2');

INSERT INTO room VALUES ('204', '2');

INSERT INTO room VALUES ('205', '2');

INSERT INTO room VALUES ('301', '3');

INSERT INTO room VALUES ('302', '3');

INSERT INTO room VALUES ('303', '3');

INSERT INTO room VALUES ('304', '3');

INSERT INTO room VALUES ('305', '3');
```

manager:

```
INSERT INTO manager VALUES ('1001', 'tjx', '1234');

INSERT INTO manager VALUES ('1002', 'dd', '4321');
```

member:

```
INSERT INTO member VALUES ('1', 'sally', 'f', '22', '08579201522');

INSERT INTO member VALUES ('2', 'john', 'm', '23', '03828383974');

INSERT INTO member VALUES ('3', 'dorothy', 'f', '30', '07478291030');

INSERT INTO member VALUES ('4', 'tom', 'm', '40', '03012029381');

INSERT INTO member VALUES ('5', 'tina', 'f', '18', '07112569881');

INSERT INTO member VALUES ('6', 'chris', 'm', '25', '07559665671');
```

guest:

```
INSERT INTO guest VALUES ('1001', 'ann', 'f', '14', 'jsi903410', '1');

INSERT INTO guest VALUES ('1002', 'andy', 'f', '23', 'ioa19342', '1');

INSERT INTO guest VALUES ('1003', 'sally', 'f', '22', 'dko12312', '1');

INSERT INTO guest VALUES ('2001', 'green', 'm', '46', 'wer12345', '2');

INSERT INTO guest VALUES ('2002', 'mary', 'f', '47', 'thc32231', '2');

INSERT INTO guest VALUES ('3001', 'dorothy', 'f', '30', 'wer12314', '3');

INSERT INTO guest VALUES ('3002', 'ming', 'm', '34', 'ioq21301', '3');

INSERT INTO guest VALUES ('4001', 'tom', 'm', '40', 'clm30123', '4');

INSERT INTO guest VALUES ('5001', 'tim', 'm', '18', 'cgg33693', '5');

INSERT INTO guest VALUES ('5002', 'tina', 'f', '18', 'oku33777', '5');

INSERT INTO guest VALUES ('6001', 'chris', 'm', '5', 'iws36548', '6');

INSERT INTO guest VALUES ('6002', 'laura', 'f', '24', 'wod21366', '6');

INSERT INTO guest VALUES ('6003', 'alex', 'm', '55', 'eaf21651', '6');

INSERT INTO guest VALUES ('6004', 'mary', 'f', '53', 'saw32645', '6');
```

order:

```
INSERT INTO orders VALUES ('1', TO_DATE('2019.1.1', 'YYYY-MM-DD'), TO_DATE('2019.1.2',
'YYYY-MM-DD'), '1', '101', '1001', '1002', '1003');

INSERT INTO orders VALUES ('2',  TO_DATE('2019.2.4', 'YYYY-MM-DD'), TO_DATE('2019.2.7',
'YYYY-MM-DD'), '2', '104', '2001', '2002', null);

INSERT INTO orders VALUES ('3',  TO_DATE('2019.3.6', 'YYYY-MM-DD'),  TO_DATE('2019.3.7',
'YYYY-MM-DD'), '3', '202', '3001', '3002', null);
```

INSERT INTO orders VALUES ('4', TO_DATE('2019.4.7', 'YYYY-MM-DD'), TO_DATE('2019.4.10', 'YYYY-MM-DD'), '3', '202', '3001', '3002', null);

INSERT INTO orders VALUES ('5', TO_DATE('2019.12.8', 'YYYY-MM-DD'), TO_DATE('2019.12.9', 'YYYY-MM-DD'), '2', '203', '3001', '3002', null);

INSERT INTO orders VALUES ('6', TO_DATE('2019.12.9', 'YYYY-MM-DD'), TO_DATE('2019.12.11', 'YYYY-MM-DD'), '4', '102', '4001', null, null);

INSERT INTO orders VALUES ('7', TO_DATE('2019.12.8', 'YYYY-MM-DD'),TO_DATE('2019.12.15', 'YYYY-MM-DD'), '5', '105', '5001', null, null);

INSERT INTO orders VALUES ('8', TO_DATE('2019.12.8', 'YYYY-MM-DD'), TO_DATE('2019.12.15', 'YYYY-MM-DD'), '5', '104', '5002', null, null);

INSERT INTO orders VALUES ('9', TO_DATE('2019.12.9', 'YYYY-MM-DD'), TO_DATE('2019.12.13', 'YYYY-MM-DD'), '6', '301', '6001', '6002', null);

INSERT INTO orders VALUES ('10', TO_DATE('2019.12.9', 'YYYY-MM-DD'), TO_DATE('2019.12.13', 'YYYY-MM-DD'), '6', '302', '6003', '6004', null);

INSERT INTO orders VALUES ('11', TO_DATE('2019.4.1', 'YYYY-MM-DD'), TO_DATE('2019.4.13', 'YYYY-MM-DD'), '5', '305', '5001', '5002', null);

INSERT INTO orders VALUES ('12', TO_DATE('2019.7.7', 'YYYY-MM-DD'), TO_DATE('2019.7.13', 'YYYY-MM-DD'), '1', '104', '1003', null, null);

INSERT INTO orders VALUES ('13', TO_DATE('2019.9.26', 'YYYY-MM-DD'), TO_DATE('2019.10.2', 'YYYY-MM-DD'), '4', '304', '4001', null, null);

4. Views:

I. view1: **room** (room_id, type name, type price)

EXPLAINATIONS:

The view ROOM is created for the use of MEMBERs when they tried place an order. MEMBERs can choose the specific room, and prices of different types varies by seasons. For instance, in the peak season summer, the price would be like what shown in the display, but in winter, the price will drop.

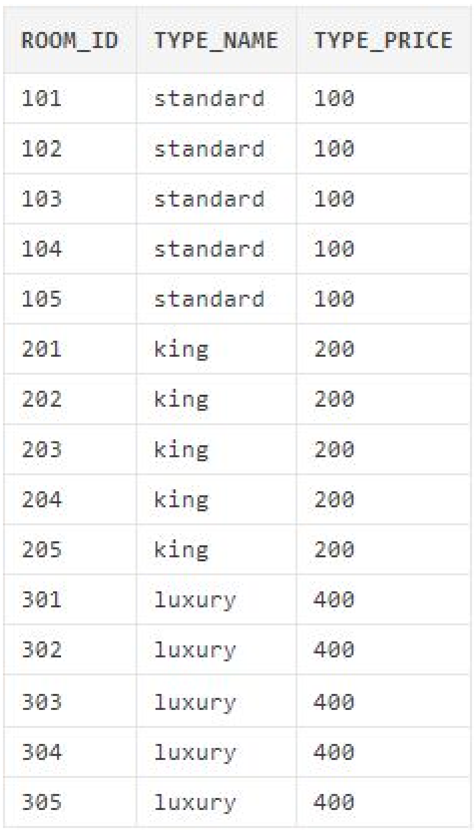
This view can also be seen by the front desk staff when guests just pop in the front desk and ask to check in an unbooked room right now.

The view combines some information in TABLE ROOM and TABLE ROOM_TYPE, makes it simpler for the non-professional people like MEMBERs and front desk staff to get useful information without any unnecessary difficulties.

SQL COMMANDS:

```
create view room_view as select room_id, type_name, type_price from room, room_type

where room.room_type_id= room_type.Type_id;

SELECT *

FROM room_view;
```

DISPLAY:



ROOM_ID	TYPE_NAME	TYPE_PRICE
101	standard	100
102	standard	100
103	standard	100
104	standard	100
105	standard	100
201	king	200
202	king	200
203	king	200
204	king	200
205	king	200
301	luxury	400
302	luxury	400
303	luxury	400
304	luxury	400
305	luxury	400

II. view2: **check-in**(order_id, check in, check out, book name, order_room_id, number of guests)

EXPLANATIONS:

The view is created for the frontdesk staff. When the GUESTS went to the front desk of the hotel and tell the staff a room/rooms have been booked by a name. The staff then can check the view by search names the guests provided.

On one side, the view is for security for there would be some sensitive information like the verification id of the guests, and it is neither secure enough nor complied with the data privacy law (GDPR) to give the staff access to the sensitive data.

On another side, the view will also simplify the querying process of booking details.

SQL COMMANDS:

create view check_in_view as SELECT order_id, Check_In, Check_Out, name as book_name, order_ROOM_id,

(case when order_GUEST3_id>0 then 1 else 0 end) + (case when order_GUEST2_id>0 then 1 else 0 end) + (case when order_GUEST1_id>0 then 1 else 0 end)

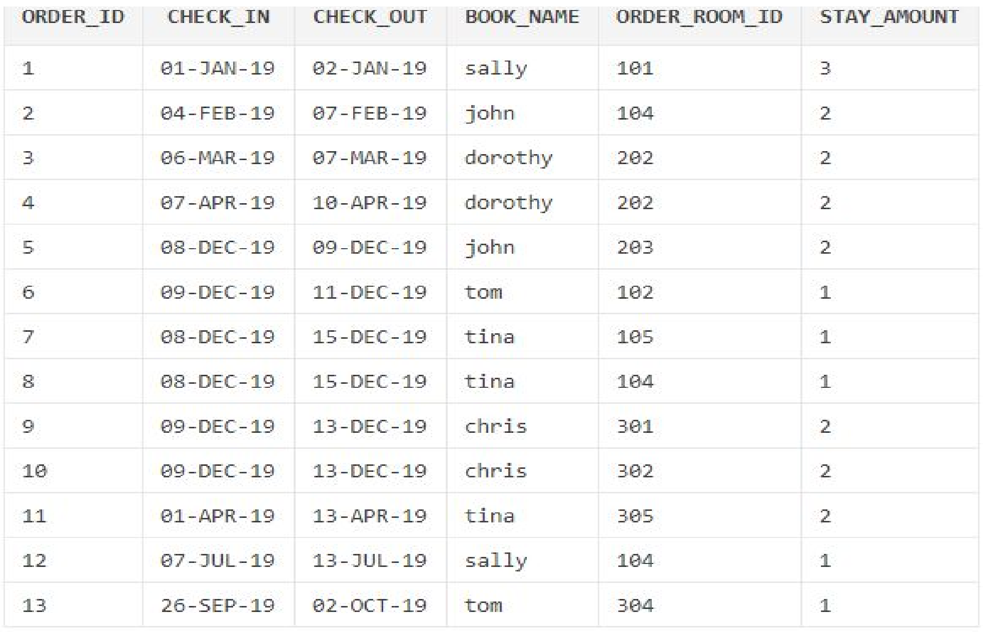
as stay_amount FROM orders, member

where orders.order_MEMBER_id = member.MEMBER_id

order by ORDER_id;

select * from check_in_view;

DISPLAY:



ORDER_ID	CHECK_IN	CHECK_OUT	BOOK_NAME	ORDER_ROOM_ID	STAY_AMOUNT
1	01-JAN-19	02-JAN-19	sally	101	3
2	04-FEB-19	07-FEB-19	john	104	2
3	06-MAR-19	07-MAR-19	dorothy	202	2
4	07-APR-19	10-APR-19	dorothy	202	2
5	08-DEC-19	09-DEC-19	john	203	2
6	09-DEC-19	11-DEC-19	tom	102	1
7	08-DEC-19	15-DEC-19	tina	105	1
8	08-DEC-19	15-DEC-19	tina	104	1
9	09-DEC-19	13-DEC-19	chris	301	2
10	09-DEC-19	13-DEC-19	chris	302	2
11	01-APR-19	13-APR-19	tina	305	2
12	07-JUL-19	13-JUL-19	sally	104	1
13	26-SEP-19	02-OCT-19	tom	304	1

III. view3: **invoice**(order_id, check in, check out, book name, total price)

EXPLANATIONS:

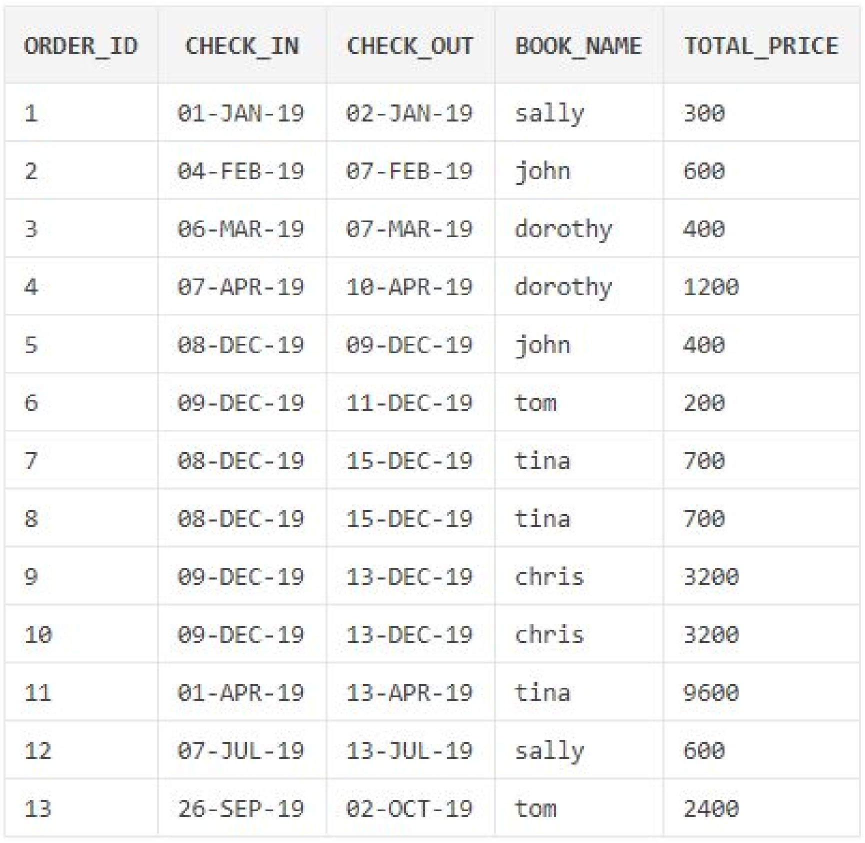
The view is created for guests. It will be used when the guests pay the bill.

For the guests, they don't need to know the information other than the price and book records. It would be safer and simpler for them to just look through the view.

SQL COMMANDS:

```
create view invoice as select order_id, Check_In, Check_Out, book_name, stay_amount *
room_view.type_price * (to_date(Check_Out) - to_date(Check_In)) as total_price from
check_in_view, room_view

where check_in_view.order_ROOM_id = room_view.room_id

order by order_id;

SELECT *

FROM invoice;
```

DISPLAY:



ORDER_ID	CHECK_IN	CHECK_OUT	BOOK_NAME	TOTAL_PRICE
1	01-JAN-19	02-JAN-19	sally	300
2	04-FEB-19	07-FEB-19	john	600
3	06-MAR-19	07-MAR-19	dorothy	400
4	07-APR-19	10-APR-19	dorothy	1200
5	08-DEC-19	09-DEC-19	john	400
6	09-DEC-19	11-DEC-19	tom	200
7	08-DEC-19	15-DEC-19	tina	700
8	08-DEC-19	15-DEC-19	tina	700
9	09-DEC-19	13-DEC-19	chris	3200
10	09-DEC-19	13-DEC-19	chris	3200
11	01-APR-19	13-APR-19	tina	9600
12	07-JUL-19	13-JUL-19	sally	600
13	26-SEP-19	02-OCT-19	tom	2400

IV. view4: **order detail**(order_id, check in, check out, book name, order_room_id, type price, number of guests, total price)——manager

EXPLANATIONS:

The view is created for the backend MANAGERS to manipulate data and financial staff such as accountants and auditors to check the bookkeeping.

By including almost all the useful information in a single view, a manager can modify and change the data easily. For the financial staff, a view which contains all the information will be more accurate and easy accessed for they are not technique professionals as well.

SQL COMMANDS:

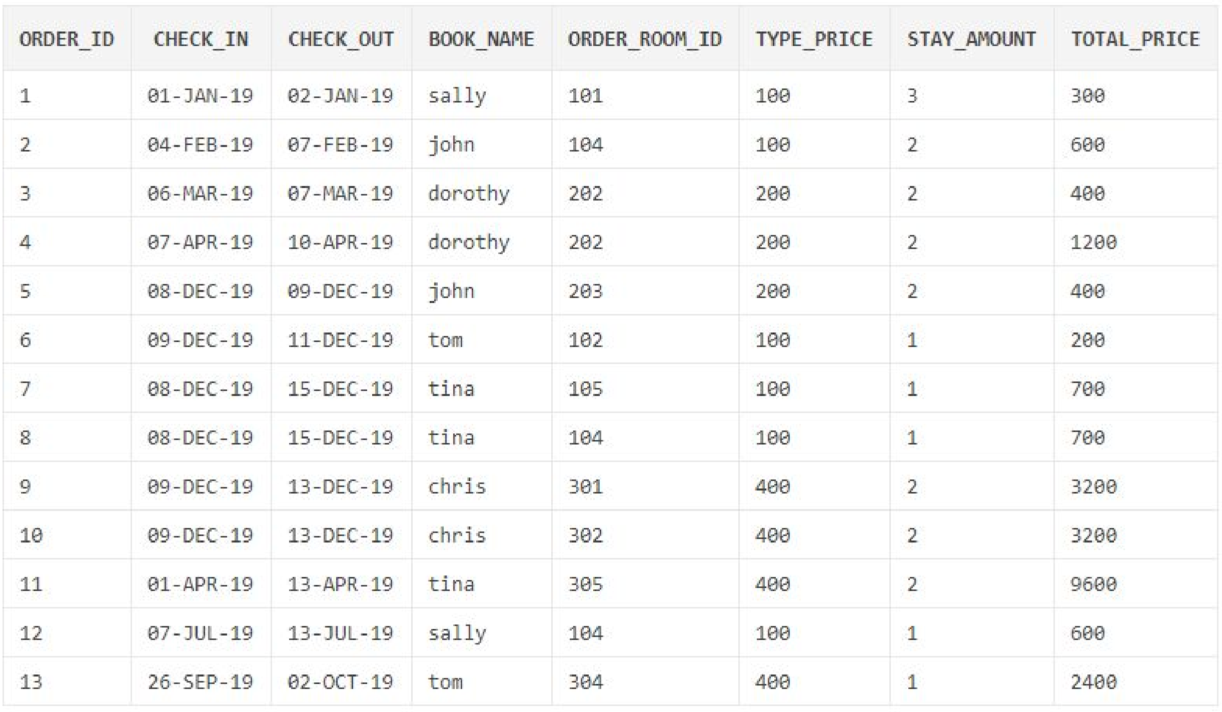
create view order_detail_view as select check_in_view.order_id, check_in_view.check_in, check_in_view.check_out, check_in_view.book_name, check_in_view.order_room_id, type_price, stay_amount, total_price from invoice, room_view, check_in_view

where check_in_view.order_ROOM_id= room_view.room_id and check_in_view.order_id=invoice.order_id;

SELECT *

FROM order_detail_view;

DISPLAY:



ORDER_ID	CHECK_IN	CHECK_OUT	BOOK_NAME	ORDER_ROOM_ID	TYPE_PRICE	STAY_AMOUNT	TOTAL_PRICE
1	01-JAN-19	02-JAN-19	sally	101	100	3	300
2	04-FEB-19	07-FEB-19	john	104	100	2	600
3	06-MAR-19	07-MAR-19	dorothy	202	200	2	400
4	07-APR-19	10-APR-19	dorothy	202	200	2	1200
5	08-DEC-19	09-DEC-19	john	203	200	2	400
6	09-DEC-19	11-DEC-19	tom	102	100	1	200
7	08-DEC-19	15-DEC-19	tina	105	100	1	700
8	08-DEC-19	15-DEC-19	tina	104	100	1	700
9	09-DEC-19	13-DEC-19	chris	301	400	2	3200
10	09-DEC-19	13-DEC-19	chris	302	400	2	3200
11	01-APR-19	13-APR-19	tina	305	400	2	9600
12	07-JUL-19	13-JUL-19	sally	104	100	1	600
13	26-SEP-19	02-OCT-19	tom	304	400	1	2400

5. Queries:

1) Who Sally has booked rooms for?

EXPLANATIONS AND PURPOSES:

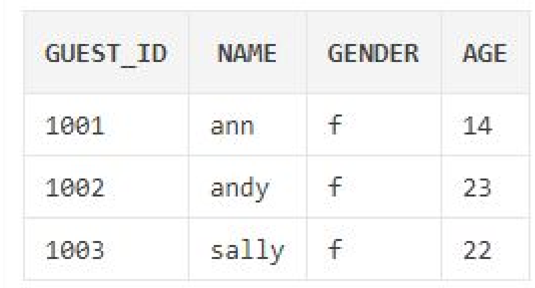
Check the booking records placed by MEMBER Sally to list all the guests the booking records included.

SQL COMMANDS:

SELECT guest.GUEST_id, guest.Name, guest.gender, guest.age from guest join member on guest.GUEST_MEMBER_id=member.MEMBER_id

where member.Name='sally';

DISPLAY:



GUEST_ID	NAME	GENDER	AGE
1001	ann	f	14
1002	andy	f	23
1003	sally	f	22

----

2) Check the availability of room 104 on 2019.12.8.

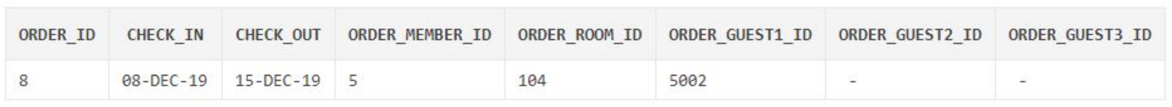
EXPLANATIONS AND PURPOSES:

Make queries to check is there any existing booking record included the day 8th Dec. 2019. If there was an existing order, room 104 is not available, vice versa.

SQL COMMANDS:

SELECT * FROM orders where order_ROOM_id=104 and Check_in<='08-DEC-19' and Check_Out>'08-DEC-19'

DISPLAY:



ORDER_ID	CHECK_IN	CHECK_OUT	ORDER_MEMBER_ID	ORDER_ROOM_ID	ORDER_GUEST1_ID	ORDER_GUEST2_ID	ORDER_GUEST3_ID
8	08-DEC-19	15-DEC-19	5	104	5002	-	-

---

3) Check how many orders have been placed in total

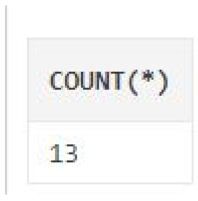
EXPLANATIONS AND PURPOSES:

Calculate the number of rows from table orders.

SQL COMMANDS:

Select count(*) from orders

DISPLAY:



COUNT(*)
13

---

4) Check which rooms has been reserved on 2019.12.9

EXPLANATIONS AND PURPOSES:

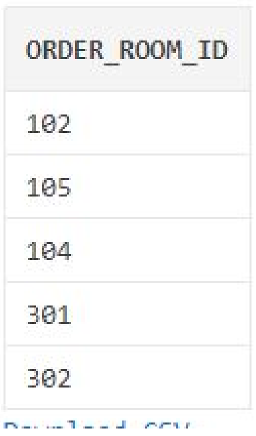
Check table orders to list all the room id of the orders whose check in dates are no later than 9th Dec. 2019 and check out dates are later than that day.

SQL COMMANDS:

SELECT order_ROOM_id FROM orders

where Check_in<='09-DEC-19' and Check_Out>'09-DEC-19'

DISPLAY:



ORDER_ROOM_ID
102
105
104
301
302

----

5) Calculate the proportion of each room type of all the orders

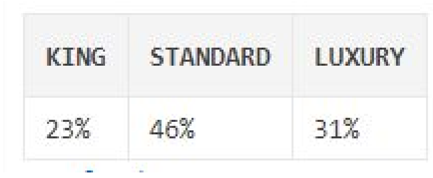
EXPLANATIONS AND PURPOSES:

Calculate the percentage of three types from view 4 “order details”.

SQL COMMANDS:

```
Select round(sum(case when type_name='king' then 1 else 0 end )/count(*)*100) ||'%' as king, round(sum(case when type_name= 'standard' then 1 else 0 end)/count(*)*100) ||'%' as standard, round(sum(case when type_name='luxury' then 1 else 0 end)/count(*)*100) ||'%' as luxury from room_view, orders

 where room_view.room_id=orders.order_room_id;
```

DISPLAY:



KING	STANDARD	LUXURY
23%	46%	31%

----

6) Check the orders taken by john.

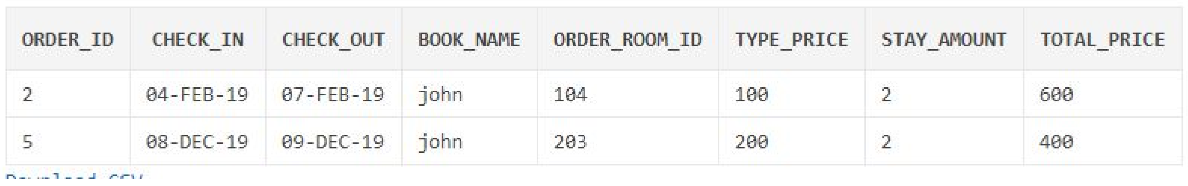
EXPLANATIONS AND PURPOSES:

Check order detail view for the orders placed by MEMBER john.

SQL COMMANDS:

```
SELECT * FROM ORDER_DETAIL_VIEW

where book_name='john'
```

DISPLAY:



ORDER_ID	CHECK_IN	CHECK_OUT	BOOK_NAME	ORDER_ROOM_ID	TYPE_PRICE	STAY_AMOUNT	TOTAL_PRICE
2	04-FEB-19	07-FEB-19	john	104	100	2	600
5	08-DEC-19	09-DEC-19	john	203	200	2	400

----

7) Check how long order 2(id) has booked.

EXPLANATIONS AND PURPOSES:

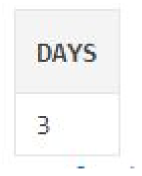
Make a query to calculate the days between check in and check out date of order 2.

SQL COMMANDS:

SELECT (to_date(Check_Out) - to_date(Check_In)) as days FROM orders

where order_id=2

DISPLAY:



DAYS
3

----

8) Make a query to check how many people order 3 has included.

EXPLANATIONS AND PURPOSES:

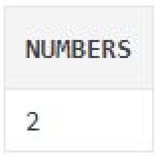
Check the three guest attributes to see if it’s null. If one guest attribute is null, count as 0, if not, count as 1, together with a condition that id equals 3 from table orders. Add them all.

SQL COMMANDS:

SELECT (case when order_GUEST3_id>0 then 1 else 0 end) + (case when order_GUEST2_id>0 then 1 else 0 end) + (case when order_GUEST1_id>0 then 1 else 0 end) as numbers FROM orders

where order_id=3

DISPLAY:



NUMBERS
2

----

9) Calculate the proportion of female and male guests

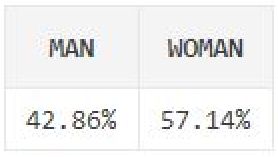
EXPLANATIONS AND PURPOSES:

Calculate the percentage of guests whose ‘gender’ equals to ‘m’ and ‘f’.

SQL COMMANDS:

SELECT round(sum(case when gender='m' then 1 else 0 end)/count(*)*100,2)||'%' as man,round(sum(case when gender='f' then 1 else 0 end)/count(*)*100,2)||'%' as woman FROM guest ;

DISPLAY:



MAN	WOMAN
42.86%	57.14%

----

10) Make a query to calculate the proportion of guests by age

EXPLANATIONS AND PURPOSES:

Sum up the guests whose ‘age’ is within a specific age group for table guest.

SQL COMMANDS:

SELECT sum(case when age<=12 and age>0 then 1 else 0 end) as children,

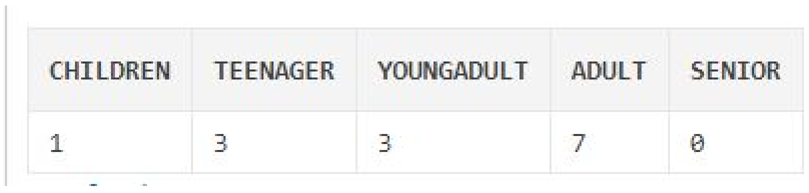
sum(case when age<=18 and age>=13 then 1 else 0 end) as teenager,

sum(case when age<=26 and age>18 then 1 else 0 end) as youngadult,

sum(case when age<=60 and age>=27 then 1 else 0 end) as adult,

sum(case when age>60 then 1 else 0 end) as senior from guest;

DISPLAY:



CHILDREN	TEENAGER	YOUNGADULT	ADULT	SENIOR
1	3	3	7	0

----

11) Check how much order 3 need to pay

EXPLANATIONS AND PURPOSES:

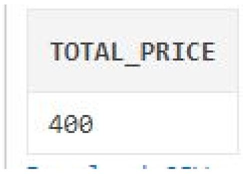
Calculate the total price based on the amount of stayed guests, type price and days stayed from the view room.

SQL COMMANDS:

select ((case when order_GUEST3_id>0 then 1 else 0 end) + (case when order_GUEST2_id>0 then 1 else 0 end) + (case when order_GUEST1_id>0 then 1 else 0 end)) * type_price * (to_date(Check_Out) - to_date(Check_In)) as total_price from room_view,orders

where order_id=3 and room_view.room_id=orders.ORDER_ROOM_ID

DISPLAY:



TOTAL_PRICE
400

---

12) Query the order amount by months

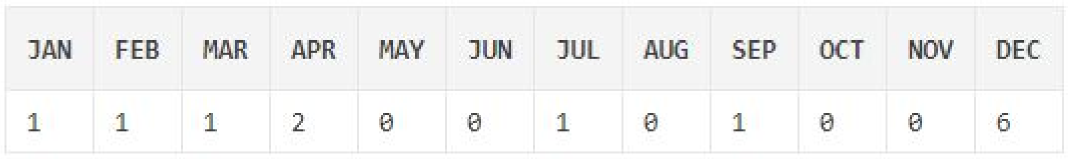
EXPLANATIONS AND PURPOSES:

Sum up the number of orders based on the ‘check in’ attribute.

SQL COMMANDS:

```
Select

sum(case when check_in like '%JAN%' then 1 else 0 end) as JAN,

sum(case when check_in like '%FEB%' then 1 else 0 end) as FEB,

sum(case when check_in like '%MAR%' then 1 else 0 end) as MAR,

sum(case when check_in like '%APR%' then 1 else 0 end) as APR,

sum(case when check_in like '%MAY%' then 1 else 0 end) as MAY,

sum(case when check_in like '%JUN%' then 1 else 0 end) as JUN,

sum(case when check_in like '%JUL%' then 1 else 0 end) as JUL,

sum(case when check_in like '%AUG%' then 1 else 0 end) as AUG,

sum(case when check_in like '%SEP%' then 1 else 0 end) as SEP,

sum(case when check_in like '%OCT%' then 1 else 0 end) as OCT,

sum(case when check_in like '%NOV%' then 1 else 0 end) as NOV,

sum(case when check_in like '%DEC%' then 1 else 0 end) as DEC from orders;
```

DISPLAY:



JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1	1	1	2	0	0	1	0	1	0	0	6