SQL Used in the Project & Project Report

姚允成 Peter Yao yy4108

Bonus

1. Used Prepared Statements to Prevent SQL Injection
2. Used WTForms Package to further enhance that the data input type from the HTML form is correct
3. Used Boostrap to beautify the UI
4. Used Google Charts to render interactive bar and pie charts
5. Used SQL procedures to provide abstraction for business logic
6. User Friendliness Provided Through additional Business Logic
    - Used Additional SQL Procedures to provide many drop down Table so that users don't need to manually input the data
    - users no longer need to enter the full information to fetch flights, can query flights based on source/destination cities OR date ALONE
    - in the drop down table of the source/destination airport, used addition sql to fetch the city information, besides the default airport code

Stored Procedures Used

agentPurchase

get the next available ticket id insert into the database the ticket that the agent has just purchased

```
delimiter //
create procedure agentPurchase(
    in airlineName  varchar(30),
    in flightNum    varchar(30),
    in bookingAgentEmail    varchar(30),
    in customerEmail    varchar(30)
)
begin
    declare nextTicketId int;

    select max(ticket_id)+1
    into nextTicketId
    from ticket;

    insert into ticket VALUES (nextTicketId, airlineName, flightNum,
bookingAgentEmail, customerEmail);
end//
delimiter ;
```

agentPurchaseConfirm

at the /agent_purchase/<airline_name>/<flight_num>

used to check at the confirmation page of flight purchase if there is any seat left at the flight that you want to puchase

```
delimiter //
create procedure agentPurchaseConfirm(
    in airlineName varchar(30),
    in flightNum varchar(30),
    in agentEmail   varchar(30)
)
begin
    declare seatsTaken int;
    declare totalSeats int;
    declare airplaneId int;

    SELECT airplane_id into airplaneId
    from flight
    where flight_num = flightNum and airline_name = airlineName;

    select seats
    into totalSeats
    from airplane
    WHERE id = airplaneId and airline_name = airlineName;

    SELECT count(*)
    into seatsTaken
    from ticket t
    where t.airline_name = airlineName
    and t.flight_num = flightNum;

    if totalSeats > seatsTaken and airlineName in (select airline_name
from works_for where agentEmail = booking_agent_email) then
        select * from flight f1 where f1.airline_name = airlineName and
f1.flight_num = flightNum;
    end if;
end//
delimiter ;
```

agentSearchNoDate

used to query all upcoming flights for the agent, if only the source and destination is provided, no need to enter the date information

also checked if there is any seat left on the flight

```
delimiter //
create procedure agentSearchNoDate(
    in depart varchar(30),
    in arrive varchar(30),
```

```
    in agentEmail varchar(30)
)
begin
    with avail_airlines as(
        select airline_name
        from works_for
        where booking_agent_email = agentEmail
    )
    select airline_name, flight_num, departure_time, arrival_time, price,
status, airplane_id, concat(arrive_airport, '/', (select city from airport
where name = arrive_airport)), concat(depart_airport, '/', (select city
from airport where name = depart_airport))
    from flight as f
    where f.airline_name in (select * from avail_airlines)
    and f.departure_time > NOW()
    and f.arrive_airport = arrive
    and f.depart_airport = depart
    and (select count(*) from ticket t where t.flight_num = f.flight_num)
<
        (select seats from airplane a where a.id = f.airplane_id and
a.airline_name = f.airline_name);
end//
delimiter ;
```

agentSearchWithDate

used to query all upcoming flights for the agent, if all the information is provided, including time Also checked if there is any seat left on the flight, if no seat left, don't show that flight

```
delimiter //
create procedure agentSearchNoDate(
    in depart varchar(30),
    in arrive varchar(30),
    in agentEmail varchar(30)
)
begin
    with avail_airlines as(
        select airline_name
        from works_for
        where booking_agent_email = agentEmail
    )
    select airline_name, flight_num, departure_time, arrival_time, price,
status, airplane_id, concat(arrive_airport, '/', (select city from airport
where name = arrive_airport)), concat(depart_airport, '/', (select city
from airport where name = depart_airport))
    from flight as f
    where f.airline_name in (select * from avail_airlines)
    and f.departure_time > NOW()
    and f.arrive_airport = arrive
    and f.depart_airport = depart
    and (select count(*) from ticket t where t.flight_num = f.flight_num)
```

```
<
        (select seats from airplane a where a.id = f.airplane_id and
a.airline_name = f.airline_name);
end//
delimiter ;
```

all_flights_taken

/view_frequent_customer/<customer_email> Used to query all the flights that a particular customer has taken with that airline

```
delimiter //
create procedure  all_flights_taken(
    in customerEmail    varchar(30),
    in airlineName      varchar(30)
)
begin
    select flight_num, date(departure_time), date(arrival_time), price,
concat(arrive_airport, "/", (select city from airport where name =
arrive_airport)), concat(depart_airport, "/", (select city from airport
where name = depart_airport))
    from ticket natural join flight
    where customer_email = customerEmail and airline_name = airlineName
    order by departure_time desc;
end//
delimiter ;
```

avail_booking_agent

query all the booking agents that work for a particular airline

```
delimiter //
create procedure avail_booking_agent(
    in airlineName  varchar(30)
)
begin
    select b.email from booking_agent b where b.email not in (
        select booking_agent_email
        from works_for
        where airline_name = airlineName
    ) ;
end//
delimiter ;
```

check_duplicate

Used in the registration phase, check if there is any existing customer/agent/staff with the same PK Prevent Insert Error

```
delimiter //
create procedure check_duplicate(
    in role varchar(30),
    in pk   varchar(30)
)
begin
    if role = "airline_staff" then
        select * from airline_staff where username = pk;
    elseif role = "customer" then
        select * from customer where email = pk;
    else
        select * from booking_agent where email = pk;
    end if;
end//
delimiter ;
```

check_duplicate_airplane

Used in add_airplane function check if the airplane id already used in the airline that the staff is working for

```
**delimiter //
create procedure check_duplicate_airplane(
    in airlineName  varchar(30),
    in planeId      int
)
begin
    select id from airplane where airline_name = airlineName and id =
planeId;
end //
delimiter ;
```

check_duplicate_airport

Used to prevent duplicate airport from being added

```
delimiter //
create procedure check_duplicate_airport(
    in airport_name varchar(6)
)
begin
    select * from airport where name = airport_name;
end//
delimiter ;
```

check_duplicate_flight

Used to prevent duplicate flights from being added

```
delimiter //
create procedure check_duplicate_flight(
    in airlineName     varchar(30),
    in flightNum       varchar(30)
)
begin
    select *
    from flight
    where airline_name = airlineName and flight_num = flightNum;
end//
delimiter ;
```

check_staff_role

Used to check the staff permission level

```
delimiter //
CREATE procedure check_staff_role(
    in user_name     varchar(30),
    in role         varchar(30)
)
begin
    select username
    from staff_permission
    where username = user_name and permit_to_do = role;
end //
delimiter ;
```

comparisonRevenueEarned

Used to get the information of revenue from direct sales and agent sales for a particular airline

```
delimiter //
create procedure comparisonRevenueEarned(
    in airlineName  varchar(30),
    out directSalesMonth    float,
    out directSalesYear     float,
    out totalSalesMonth     float,
    out totalSalesYear      float
)
begin
    select sum(price) into totalSalesMonth
    from flight natural JOIN ticket
    where airline_name = airlineName and departure_time BETWEEN
```

```
DATE_SUB(NOW(), interval 1 month) and now();

    select sum(price) into totalSalesYear
    from flight natural JOIN ticket
    where airline_name = airlineName and departure_time BETWEEN
DATE_SUB(NOW(), interval 1 YEAR) and now();

    select sum(price) into directSalesMonth
    from flight natural JOIN ticket
    where airline_name = airlineName and departure_time BETWEEN
DATE_SUB(NOW(), interval 1 month) and now() and booking_agent_email is
NULL;

    select sum(price) into directSalesYear
    from flight natural JOIN ticket
    where airline_name = airlineName and departure_time BETWEEN
DATE_SUB(NOW(), interval 1 year) and now() and booking_agent_email is
NULL;
end//
delimiter ;
```

frequent_customer

Used to see the most frequent customer for an airline within a certain timeframe

```
delimiter //
create procedure frequent_customer(
    in airlineName  varchar(30)
)
begin
    select customer_email, count(ticket_id) as total
    from flight natural join ticket
    where airline_name = airlineName
    and departure_time between DATE_SUB(now(), INTERVAL 1 year) and now()
    group by customer_email
    order by total desc;
end//
delimiter ;
```

getAirportCity

Get all the airports within the system used whenever a dropdown list of all airports are needed

```
delimiter //
create procedure getAirportCity()
begin
    select * from airport;
end //
delimiter ;
```

getCompany

Stored, but never used, strange

```
delimiter //
create procedure getCompany(
    in agentEmail varchar(30)
)
begin
    select airline_name from works_for where booking_agent_email =
agentEmail;
end //
delimiter ;
```

get_ailrine_airplane

get all the airplanes for a certain airline Used when creating a new flight, and we need to create a dropdown list of all the airplanes for that airline

```
delimiter //
create procedure get_airline_airplane(
    in airline  varchar(30)
)
begin
    select id, seats
    from airplane
    where airline_name = airline;
end//
delimiter ;
```

get_all_permission

used in granting the permisson, create a dropdown lists for all permissions that exist in the database

```
delimiter //
create procedure get_all_permission()
begin
    select * from permission;
end //
delimiter ;
```

get_all_staff

used in granting new permissions to staff, to create a drop down list for all the employees working for a specific airline, so that the admin can choose and pick

```
delimiter //
create procedure get_all_staff(
    in airlineName  varchar(30)
)
begin
    select username, first_name, last_name
    from airline_staff
    where airline_name = airlineName;
end//
delimiter ;
```

grant_new_permission

insert new permission into the db, ignore the request if the row already exists in the staff_permission table

```
delimiter //
create procedure grant_new_permission(
    in userName varchar(30),
    in newPermission   varchar(30)
)
begin
    insert ignore into staff_permission values (userName, newPermission);
end //
delimiter ;
```

insert_new_staff

as the name suggests, this is used in the staff registration process. Note that I guarantee that there cannot exist the case when the two staff share the same username, this case has already been eliminated by the corresponding check SQL.

```
delimiter //
create procedure insert_new_staff(
    in username    varchar(30),
    in password     varchar(32),
    in first_name   varchar(15),
    in last_name    varchar(15),
    in date_of_birth    date,
    in airline_name varchar(30)
)
begin
    insert into airline_staff(username, password, first_name, last_name,
date_of_birth, airline_name) values(username, md5(password), first_name,
last_name, date_of_birth, airline_name);
end//
delimiter ;
```

purchase_ticket

this procedure is stored but never used in the project!

```
delimiter //
create procedure
    flight.purchase_ticket(in arrive_airport varchar(6), in depart_airport
varchar(6), in departure_date date, out airline_name varchar(20), out
flight_num varchar(20), out departure_time datetime, out arrival_time
datetime, out price numeric(10, 2))
begin
    with avail_airplane_id as(
        select flight_num, airplane_id, seats
        from flight f join airplane a on(a.id = f.airplane_id)
        where f.depart_airport = depart_airport and
f.arrive_airport=arrive_airport and date(departure_time) = departure_date
    ),
    seats_taken as(
        select flight_num, count(customer_email) as taken
        from ticket natural join avail_airplane_id
        group by flight_num
    )
    select airline_name, flight_num, departure_time, arrival_time, price
    from flight natural join avail_airplane_id natural join seats_taken
    where avail_airplane_id.seats - seats_taken.taken > 0;
end //
```

staff_view_booking_agent

this is a proc to query best agents for a certain airline based on multiple benchmarks. Just pass different parameters into the procedure

```
delimiter //
create procedure staff_view_booking_agent(
    in airlineName  varchar(30),
    in benchmark    varchar(30),
    in dateRange    varchar(30)
)
begin
    if benchmark="ticket" and dateRange="year" then
        select booking_agent_email ,count(*) as total
        from ticket natural join flight
        where booking_agent_email in (
            select w.booking_agent_email from works_for as w where
w.airline_name = airlineName
        ) and (departure_time between DATE_SUB(NOW(), INTERVAL 1 YEAR) and
NOW())
        group by booking_agent_email
        order by total desc
        limit 5;
```

```
    elseif benchmark = "ticket" and dateRange = "month" then
        select booking_agent_email, count(*) as total
        from ticket natural join flight
        where booking_agent_email in (
            select w.booking_agent_email from works_for as w where
w.airline_name = airlineName
        ) and (departure_time between DATE_SUB(NOW(), INTERVAL 1 MONTH)
and NOW())
        group by booking_agent_email
        order by total desc
        limit 5;
    elseif benchmark = "amount" and dateRange = "year" then
        select booking_agent_email ,sum(price) as total
        from ticket natural join flight
        where booking_agent_email in (
            select w.booking_agent_email from works_for as w where
w.airline_name = airlineName
        ) and (departure_time between DATE_SUB(NOW(), INTERVAL 1 YEAR) and
now())
        group by booking_agent_email
        order by total desc
        limit 5;
    end if;
end //
delimiter ;
```

staff_view_flights_city

this proc is for staff to query all upcoming flights for the airline they work for, based on the source and destination cities.

```
delimiter //
create procedure staff_view_flights_city (
    in airlineName  varchar(30),
    in sourceCity   varchar(30),
    in destinationCity  varchar(30)
)
begin
    select airline_name, flight_num, departure_time, arrival_time, price,
status, airplane_id, concat(arrive_airport, '/', (select city from airport
where name = arrive_airport)), concat(depart_airport, '/', (select city
from airport where name = depart_airport))
    from flight
    where airline_name = airlineName and depart_airport = sourceCity and
arrive_airport = destinationCity and departure_time > now();
end//
delimiter ;
```

staff_view_flights_date

this proc is for staff to query the flight of the airlines they work for, based on the date only, increase flexibility

```
delimiter //
create procedure staff_view_flights_date (
    in airlineName  varchar(30),
    in startDate    date,
    in endDate      date
)
begin
    select airline_name, flight_num, departure_time, arrival_time, price,
status, airplane_id, concat(arrive_airport, '/', (select city from airport
where name = arrive_airport)), concat(depart_airport, '/', (select city
from airport where name = depart_airport))
    from flight
    where airline_name = airlineName and departure_time between startDate
and endDate;
end//
delimiter ;
```

staff_view_flights_date_city

this proc is for staff to query the flight of the airlines they work for, based on the date and cities, namely, when all info is provided

```
delimiter //
create procedure staff_view_flights_date_city (
    in airlineName  varchar(30),
    in startDate    date,
    in endDate      date,
    in sourceCity   varchar(30),
    in destinationCity  varchar(30)
)
begin
    select airline_name, flight_num, departure_time, arrival_time, price,
status, airplane_id, concat(arrive_airport, '/', (select city from airport
where name = arrive_airport)), concat(depart_airport, '/', (select city
from airport where name = depart_airport))
    from flight
    where airline_name = airlineName and depart_airport = sourceCity and
arrive_airport = destinationCity and date(departure_time) between
startDate and endDate;
end//
delimiter ;
```

staff_view_flights_default

default query when the staff loads the view flight page, show all flights coming up in the next 30 days

```
delimiter //
create procedure staff_view_flights_default (
    in airlineName varchar(30)
)
begin
    select airline_name, flight_num, departure_time, arrival_time, price,
status, airplane_id, concat(arrive_airport, '/', (select city from airport
where name = arrive_airport)), concat(depart_airport, '/', (select city
from airport where name = depart_airport))
    from flight
    where airline_name = airlineName and departure_time between now() and
date_add(now(), interval 30 day);
end//
delimiter ;
```

top_destination_1_year

query the top destinations for that airline in the past year

```
delimiter //
create procedure top_destination_1_year(
    in airlineName  varchar(30)
)
begin
    with destination as (
        select count(ticket_id) as total, arrive_airport
        from ticket natural join flight
        where airline_name = airlineName and departure_time between
DATE_SUB(date(now()), interval 1 YEAR) and now()
        group by arrive_airport
        order by total desc
        limit 3
    )
    select concat(city, "/", arrive_airport), total
    from destination join airport on (destination.arrive_airport =
airport.name);
end//
delimiter ;
```

top_destination_3_month

similar to the proc above, but limit the time frame to the last 3 months

```
delimiter //
create procedure top_destination_3_month(
    in airlineName  varchar(30)
)
begin
```

```
    with destination as (
        select count(ticket_id) as total, arrive_airport
        from ticket natural join flight
        where airline_name = airlineName and departure_time between
DATE_SUB(date(now()), interval 3 MONTH) and now()
        group by arrive_airport
        order by total desc
        limit 3
    )
    select concat(city, "/", arrive_airport), total
    from destination join airport on (destination.arrive_airport =
airport.name);
end//
delimiter ;
```

top_five_commission_last_year

this proc is used to query the top customers last year of an agent based on the total amount of commission received

```
delimiter //
create procedure top_five_commission_last_year(
    agentEmail  varchar(30)
)
begin
    select name, customer_email, sum(f.price) * 0.1 as commission
    from (ticket t natural join flight f) join customer c on
t.customer_email = c.email
    where t.booking_agent_email = agentEmail and departure_time BETWEEN
DATE_SUB(now(), INTERVAL 1 YEAR) and now()
    group by customer_email, c.name
    order by commission DESC
    limit 5;
end//
delimiter ;
```

top_five_ticket_count

this show the top 5 customers of a particular agent last 6 months, based on the number of tickets sold

```
delimiter //
create procedure top_five_ticket_count(
    in agentEmail   varchar(30)
)
begin
    select name, customer_email, count(ticket_id) as ticket_count
    from (ticket t natural join flight f) join customer c on
t.customer_email = c.email
    where t.booking_agent_email = agentEmail and departure_time BETWEEN
```

```
DATE_SUB(now(), INTERVAL 6 MONTH) and now()
    group by customer_email, c.name
    order by ticket_count DESC
    limit 5;
end //
delimiter ;
```

update flight status

this proc is for the staff of an airline to update the status of a particular flight of that airline

```
delimiter //
create procedure update_flight_status(
    in airlineName     varchar(30),
    in flightNum       varchar(30),
    in new_status      varchar(30)
)
begin
    update flight
    set status = new_status
    where airline_name = airlineName and flight_num = flightNum;
end//
delimiter ;
```

viewMyCommissionNotDefault

view the commission received by a particular agent based on the start and end date he has entered into the form

```
delimiter //
create procedure viewMyCommissionNotDefault(
    in agentEmail   varchar(30),
    in startDate    date,
    in dateRange    int,
    out totalAmount numeric(12,2),
    out averageCommission   numeric(12,2),
    out totalTicket int
)
begin
    select sum(price) * 0.1
    into totalAmount
    from ticket natural join flight
    where booking_agent_email=agentEmail
    and datediff(startDate, date(departure_time)) BETWEEN 0 and dateRange;

    select count(ticket_id) into totalTicket
    from ticket natural join flight
    where booking_agent_email = agentEmail
    and datediff(startDate, date(departure_time)) BETWEEN 0 and dateRange;
```

```
    SELECT totalAmount/totalTicket into averageCommission;
end//
delimiter ;
```

view_reports

view monthly ticket sales breakdown for a particular airline based on the number of tickets sold

```
delimiter //
create procedure view_reports(
    in airlineName  varchar(30),
    in beginDate    date,
    in endDate      date
)
begin
    with temp as (
        select count(ticket_id) as total, year(departure_time) as year,
month(departure_time) as month
        from ticket natural join flight
        where airline_name = airlineName and date(departure_time) between
beginDate and endDate
        group by year(departure_time), month(departure_time)
    )
    select concat(year, "/", month) as time, total
    from temp
    order BY time;
end //
delimiter ;
```

view_reports_total

see the total number of tickets sold by an airline in a certain time frame

```
delimiter //
create procedure view_reports_total(
    in airlineName  varchar(30),
    in beginDate    date,
    in endDate      date
)
begin
    select count(ticket_id) as total
    from ticket natural join flight
    where airline_name = airlineName and date(departure_time) between
beginDate and endDate;
end //
delimiter ;
```

Plain SQL Used

> back then I didn't know how to use procedures Later, I am too lazy to update they works, but ugly

customer purchase query

```
with avail_airplane_id as(
    select flight_num, airplane_id, seats
    from flight f join airplane a on(a.id = f.airplane_id)
    where f.depart_airport = '{}' and f.arrive_airport='{}' and
date(departure_time) = '{}'
), seats_taken as(
        select flight_num, count(customer_email) as taken
        from ticket natural right outer join avail_airplane_id
        group by flight_num
    )
    select airline_name, flight_num, departure_time, arrival_time, price
    from flight natural join avail_airplane_id natural join seats_taken
    where avail_airplane_id.seats - seats_taken.taken > 0;
```

get the max ticket id in the customer purchase page

```
max_query = "select max(ticket_id) from ticket;"
```

insert the customer purchase ticket data into the db

```
insert_ticket_query = "insert into ticket values('{}', '{}', '{}', NULL,
'{}');"
cursor.execute(insert_ticket_query.format(max_ticket_id, airline_name,
flight_num, session['email']))
```

login portal for all users

```
conn.reconnect()
cursor=conn.cursor()
if role != "airline_staff":
    query="select * from {} where email='{}' and password=md5('{}');"
else:
    query = "select * from {} where username='{}' and password=md5('{}');"
cursor.execute(query.format(role, email, password))
```

show all upcoming flights for a particular customer

```
upcoming_query="select airline_name, flight_num,
DATE_FORMAT(departure_time, '%Y.%m.%d %k:%i'), DATE_FORMAT(arrival_time,
'%Y.%m.%d %k:%i'), status, arrive_airport, depart_airport from\
    flight join ticket using (airline_name, flight_num) where
customer_email = '{}' and arrival_time > curtime();"
cursor.execute(upcoming_query.format(email))
```

calculate the toal money spent last year

```
total_money_query = "select sum(price) as spending_last_year from flight
join ticket using (airline_name, flight_num) where customer_email='{}' and
departure_time between DATE_SUB(NOW(),INTERVAL 1 YEAR) and NOW();"
```

monthly breakdown of money spent by a customer

```
spending_query="select sum(price) as spending, year(departure_time) as
year, month(departure_time) as month from flight join ticket using
(airline_name, flight_num) where customer_email='{}' and departure_time
between DATE_SUB(NOW(), INTERVAL 6 MONTH) and NOW() group by
year(departure_time), month(departure_time);"
cursor.execute(spending_query.format(email))
```

monthly breakdown of money spent by a customer within a specified time window

```
spending_query="select sum(price) as spending, year(departure_time) as
year, month(departure_time) as month\
    from flight join ticket using (airline_name, flight_num)\
    where customer_email='{}' and departure_time between '{}' and '{}'\
    group by year(departure_time), month(departure_time);"
```