USE CASE STUDY REPORT

Group No: 6

Student Names: Namrata Bhartiya and Richa Talaty

I. Executive Summary

The goal of this project was to design and implement a relational database that is ready for application in a company which deals with renting of drones. This database will be beneficial in reducing data input processing time and enduring cost saving benefits. The use case of the company was examined and based on all their requirements a database was modelled. The EER and UML diagrams were designed for conceptual understanding, followed by the mapping of this conceptual model to a relational model with the identification of the required primary and foreign keys. This database was then devised using MySQL and Neo4j graph database (to study the feasibility in a NoSQL environment). The database was eventually connected to Python and Tableau to test the extent of its analytical capabilities. The next stage could be to implement data governance and security measures, in order to make this database industry ready.

II. Introduction

HiFlying Drones is a company which rents drones out to customers.

The company purchases a range of different types of drones to meet their customers' requirements. Each type of drone they purchase is assigned a drone type code (e.g., PH4) as the identifier for this type. HiFlying also records the drone types of models (e.g., Phantom 4) and the carrying capacity of the drone type in kilograms. The company assigns a training course to each type of drone they purchase which customers must successfully complete before they are permitted to rent a drone. Each training course is identified by a training code, has a training description and is for a set number of hours. A given training course may be used for several different types of drones.

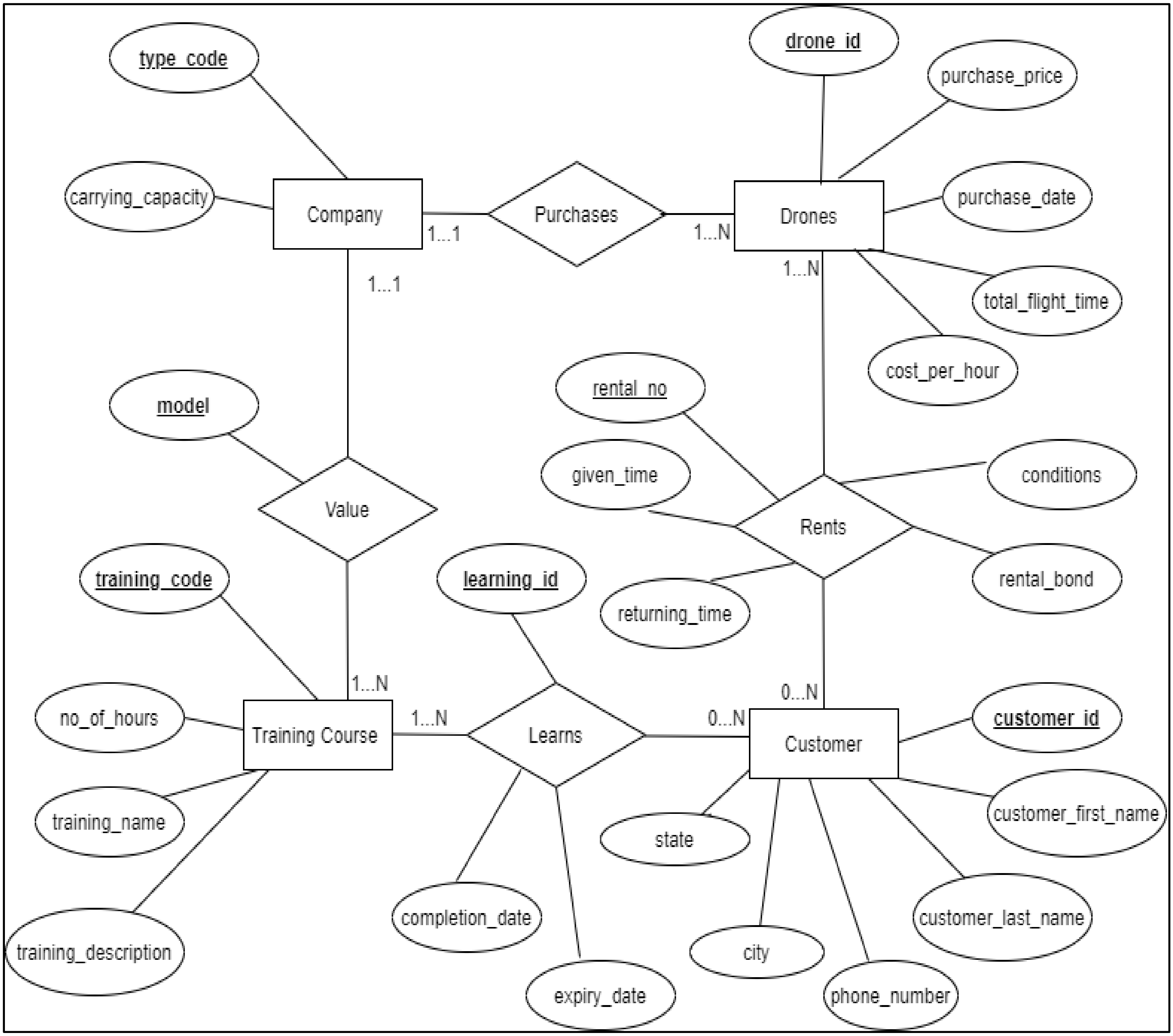
To keep track of the drones they purchase, HiFlying identifies each drone with a drone id. When a new drone is added to the system the type of the drone, the date it was purchased, and the purchase price are recorded. In addition, HiFlying establishes a drone hire rate as a cost per hour for customers to rent this drone (rates per hour are often changed over the life of the drone, as it ages, although they are only interested in recording the current cost per hour for the drone). In addition, HiFlying records the total flight time this drone has completed since purchase, this figure is updated when a customer returns a drone.

HiFlying customers are identified by a customer id; the company also record the customer's name and a contact phone number. Customers are required to be added to the HiFlying system before they can complete a training course or arrange a rental. A drone cannot be rented if the customer has not completed the appropriate training course for the type of drone they wish to rent. When a customer completes a training course, the date they completed the course is recorded and an expiry date is set, after which, if the customer wishes to continue to fly this type of drone, they will need to repeat the course.

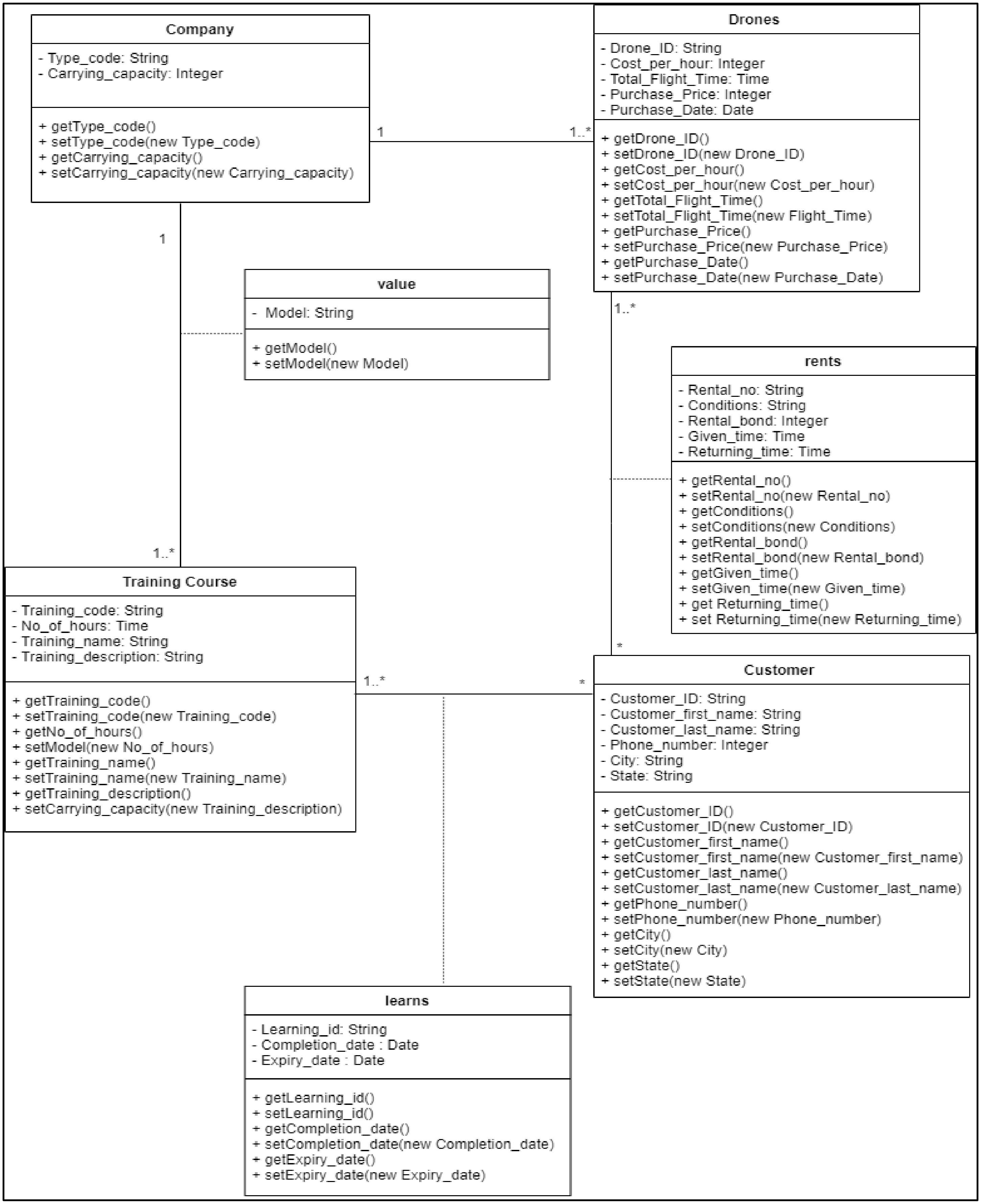
When a customer rents a particular drone, a unique rental number is assigned for the rental, the date/time that the drone leaves HiFlying is recorded, and a rental bond is assigned. As a requirement of the rental, the customer must have completed the appropriate training course for this type of drone. As part of the recording of the rental, HiFlying wishes to identify which training completed by the customer allowed them to take this drone. When a customer returns a drone the return date/time is noted and provided the drone is in good order, the rental bond will be returned to the customer.

III. Conceptual Data Modelling

Entity-Relationship (ER) Diagram



UML Diagram


Company
- Type_code: String
- Carrying_capacity: Integer
+ getType_code()
+ setType_code(new Type_code)
+ getCarrying_capacity()
+ setCarrying_capacity(new Carrying_capacity)
1
1..*
Drones
- Drone_ID: String
- Cost_per_hour: Integer
- Total_Flight_Time: Time
- Purchase_Price: Integer
- Purchase_Date: Date
+ getDrone_ID()
+ setDrone_ID(new Drone_ID)
+ getCost_per_hour()
+ setCost_per_hour(new Cost_per_hour)
+ getTotal_Flight_Time()
+ setTotal_Flight_Time(new Flight_Time)
+ getPurchase_Price()
+ setPurchase_Price(new Purchase_Price)
+ getPurchase_Date()
+ setPurchase_Date(new Purchase_Date)
1
value
- Model: String
+ getModel()
+ setModel(new Model)
1..*
rents
- Rental_no: String
- Conditions: String
- Rental_bond: Integer
- Given_time: Time
- Returning_time: Time
+ getRental_no()
+ setRental_no(new Rental_no)
+ getConditions()
+ setConditions(new Conditions)
+ getRental_bond()
+ setRental_bond(new Rental_bond)
+ getGiven_time()
+ setGiven_time(new Given_time)
+ get Returning_time()
+ set Returning_time(new Returning_time)
1..*
Training Course
- Training_code: String
- No_of_hours: Time
- Training_name: String
- Training_description: String
+ getTraining_code()
+ setTraining_code(new Training_code)
+ getNo_of_hours()
+ setModel(new No_of_hours)
+ getTraining_name()
+ setTraining_name(new Training_name)
+ getTraining_description()
+ setCarrying_capacity(new Training_description)
1..*
*
*
Customer
- Customer_ID: String
- Customer_first_name: String
- Customer_last_name: String
- Phone_number: Integer
- City: String
- State: String
+ getCustomer_ID()
+ setCustomer_ID(new Customer_ID)
+ getCustomer_first_name()
+ setCustomer_first_name(new Customer_first_name)
+ getCustomer_last_name()
+ setCustomer_last_name(new Customer_last_name)
+ getPhone_number()
+ setPhone_number(new Phone_number)
+ getCity()
+ setCity(new City)
+ getState()
+ setState(new State)
learns
- Learning_id: String
- Completion_date : Date
- Expiry_date : Date
+ getLearning_id()
+ setLearning_id()
+ getCompletion_date()
+ setCompletion_date(new Completion_date)
+ getExpiry_date()
+ setExpiry_date(new Expiry_date)

IV. Mapping Conceptual Model to Relational Model

Company (**type_code**, carrying_capacity, *model*)
Primary Key **type_code**; NULL NOT ALLOWED
Foreign Key *model* refers model in Value; NULL NOT ALLOWED

Training_Course (**training_code**, training_name, no_of_hours, training_description)
Primary Key **training_code**; NULL NOT ALLOWED

Value(**model**, *training_code*)
Primary Key **model**; NULL NOT ALLOWED
Foreign Key *training_code* refers training_code in Training_Course; NULL NOT ALLOWED

Customer (**customer_id**, customer_first_name,customer_last_name ,phone_number, city, state)
Primary Key **customer_id**; NULL NOT ALLOWED

Drones (**drone_id**, purchase_price, purchase_date, total_flight_time, cost_per_hour, *type_code*)
Primary Key **drone_id**; NULL NOT ALLOWED
Foreign Key *type_code* refers type_code in Model; NULL NOT ALLOWED

Learns (**learning_id**, *training_code*, *customer_id*, completion_date, expiry_date)
Primary Key **learning_id**; NULL NOT ALLOWED
Foreign Key *training_code* refers training_code in Training_Course; NULL NOT ALLOWED
Foreign Key *customer_id* refers customer_id in Customer; NULL NOT ALLOWED

Rents (**rental_no**,*drone_id*, *customer_id*, given_time, returning_time, rental_bond, conditions)
Primary Key **rental_no**; NULL NOT ALLOWED
Foreign Key *drone_id* refers drone_id in Drones; NULL NOT ALLOWED
Foreign Key *customer_id* refers customer_id in Customer; NULL NOT ALLOWED

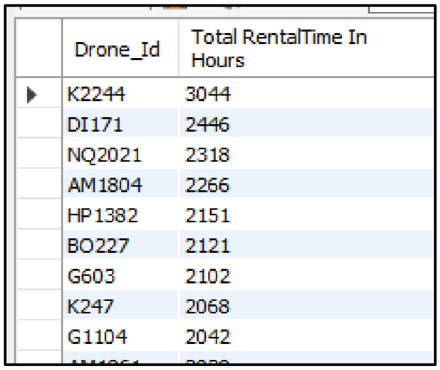
V. Implementation of Relational Model via MySQL and NoSQL

MySQL Implementation

The database was created in MySQL Workbench by implementing the relational model of the Use Case described above.

# Query 1 : Find out which drones were rented for the longest time (Top 10)

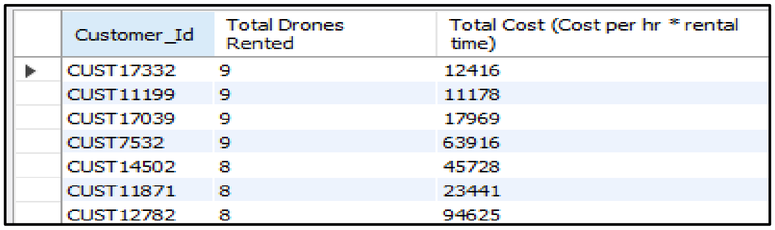
```
select r.Drone_Id,
round(sum(timestampdiff(MINUTE,r.Given_Time,r.Returning_Time))/60,0)
 as 'Total_RentalTime_In_Hours'
from Rents as r
group by r.Drone_Id
order by Total_RentalTime_In_Hours desc limit 10;
```



	Drone_Id	Total RentalTime In Hours
▶	K2244	3044
	DI171	2446
	NQ2021	2318
	AM1804	2266
	HP1382	2151
	BO227	2121
	G603	2102
	K247	2068
	G1104	2042
	[illegible]	[illegible]

#Query 2: Find the top customers with the highest rentals

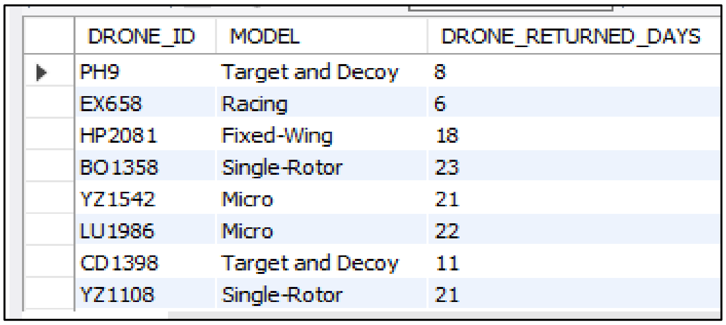
```
select r.Customer_Id, count(d.Drone_Id) as 'Total Drones Rented',
round(sum(timestampdiff(MINUTE,r.Given_Time,r.Returning_Time))/60,0)*d.Cost_Per_Hour
as 'Total Cost (Cost per hr * rental time)'
from Rents as r, Drones as d
where r.Drone_Id = d.Drone_Id
group by r.Customer_Id
order by count(d.Drone_Id) desc;
```



Customer_Id	Total Drones Rented	Total Cost (Cost per hr * rental time)
CUST17332	9	12416
CUST11199	9	11178
CUST17039	9	17969
CUST7532	9	63916
CUST14502	8	45728
CUST11871	8	23441
CUST12782	8	94625

# Query 3: Find out the drone id, model type of drones which were returned after 2 days and in bad condition

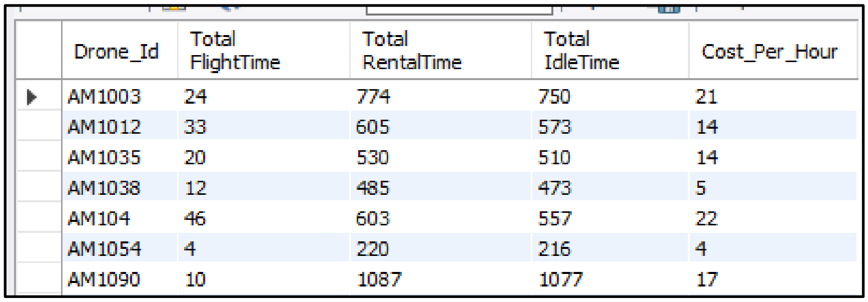
```
select distinct r.drone_id, c.model,
datediff(r.returning_time,r.given_time)
as drone_returned_days
from Rents r join Drones d
on r.drone_id = d.drone_id join company c
on d.type_code = c.type_code
where r.conditions = 'bad' and datediff(r.returning_time,r.given_time)>2;
```



DRONE_ID	MODEL	DRONE_RETURNED_DAYS
PH9	Target and Decoy	8
EX658	Racing	6
HP2081	Fixed-Wing	18
BO1358	Single-Rotor	23
YZ1542	Micro	21
LU1986	Micro	22
CD1398	Target and Decoy	11
YZ1108	Single-Rotor	21

# Query 4: Find the total flight time, rental time and idle time for each drone in hours along with the cost per hour

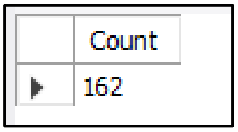
```
select d.Drone_Id, round(sum(d.Total_Flight_Time)/60,0)
as "Total FlightTime", d.Cost_Per_Hour,
round(sum(timestampdiff(MINUTE,r.Given_Time,r.Returning_Time))/60,0)
as "Total RentalTime",
round(sum((timestampdiff(MINUTE,r.Given_Time,r.Returning_Time) - d.Total_Flight_Time))/60,0)
as "Total IdleTime"
from Rents as r, Drones as d
where d.Drone_Id = r.Drone_Id
group by d.Drone_Id
order by d.Drone_Id;
```



Drone_Id	Total FlightTime	Total RentalTime	Total IdleTime	Cost_Per_Hour
AM1003	24	774	750	21
AM1012	33	605	573	14
AM1035	20	530	510	14
AM1038	12	485	473	5
AM104	46	603	557	22
AM1054	4	220	216	4
AM1090	10	1087	1077	17

# Query 5: Find the number of returning customers (customers who purchased drones every year in the last 3 years)

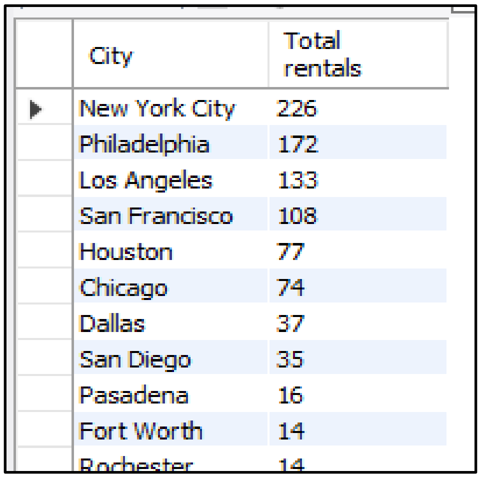
```
select count(distinct customer_id) as count from Rents
where year(given_time) = year(curdate()) and customer_id in (select customer_id from Rents
where year(given_time) = year(curdate())-1 and customer_id in (select customer_id from Rents
where year(given_time) = year(curdate())-2));
```



Count
162

# Query 6: Find the number of drone rentals from the top 5 states and cities

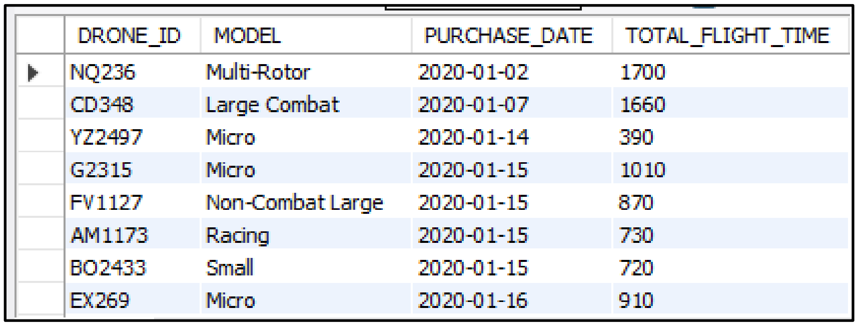
```
select c.City, count(r.rental_no) as "Total rentals"
from Customer as c, Rents as r
where c.Customer_Id = r.Customer_Id
and c.State in (select temp.State from
                (select cu.State from Customer as cu, Rents as re
                where cu.Customer_Id = re.Customer_Id
                group by cu.State
                order by count(cu.Customer_Id) desc limit 5) as temp)
group by c.City
order by count(c.Customer_Id) desc;
```



City	Total rentals
New York City	226
Philadelphia	172
Los Angeles	133
San Francisco	108
Houston	77
Chicago	74
Dallas	37
San Diego	35
Pasadena	16
Fort Worth	14
Rochester	14

# Query 7: Find out the drone id, model, purchase date, total flight time that were bought after 1st January 2020, and whose total flight time exceeded 100 hrs

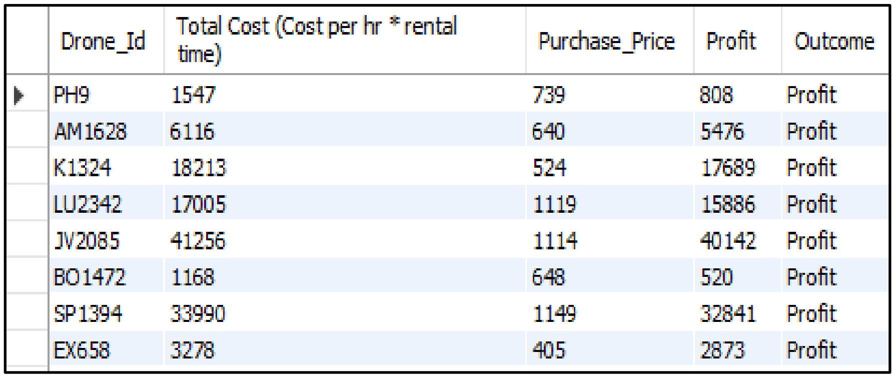
```
select d.drone_id,c.model, d.purchase_date,d.total_flight_time
from Drones d, Company c
where d.type_code = c.type_code
and d.purchase_date>'2020-01-01'
and d.total_flight_time>100
order by purchase_date asc,
total_flight_time desc;
```



DRONE_ID	MODEL	PURCHASE_DATE	TOTAL_FLIGHT_TIME
NQ236	Multi-Rotor	2020-01-02	1700
CD348	Large Combat	2020-01-07	1660
YZ2497	Micro	2020-01-14	390
G2315	Micro	2020-01-15	1010
FV1127	Non-Combat Large	2020-01-15	870
AM1173	Racing	2020-01-15	730
BO2433	Small	2020-01-15	720
EX269	Micro	2020-01-16	910

# Query 8: Find the profit earned for each drone

```
select r.Drone_Id, d.Purchase_Price,
round(sum(timestampdiff(MINUTE,r.Given_Time,r.Returning_Time))/60,0)*d.Cost_Per_Hour
as 'Total Cost (Cost per hr * rental time)',
round(sum(timestampdiff(MINUTE,r.Given_Time,r.Returning_Time))/60,0)*
d.Cost_Per_Hour - d.Purchase_Price as Profit,
CASE
WHEN round(sum(timestampdiff(MINUTE,r.Given_Time,r.Returning_Time))/60,0)*
 d.Cost_Per_Hour -  d.Purchase_Price  > 0
THEN 'Profit'
 ELSE 'Loss'
END AS Outcome
from Rents as r, Drones as d
where r.Drone_Id = d.Drone_Id
group by r.Drone_Id;
```



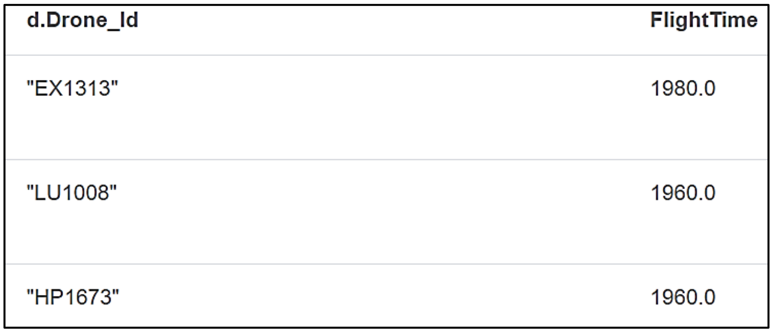
Drone_Id	Total Cost (Cost per hr * rental time)	Purchase_Price	Profit	Outcome
PH9	1547	739	808	Profit
AM1628	6116	640	5476	Profit
K1324	18213	524	17689	Profit
LU2342	17005	1119	15886	Profit
JV2085	41256	1114	40142	Profit
BO1472	1168	648	520	Profit
SP1394	33990	1149	32841	Profit
EX658	3278	405	2873	Profit

<u>NoSQL Implementation:</u>

The tables were created in the Neo4j online console and the following Cypher queries were executed.

# Query 1: Find the Drones with the longest flight time in hours

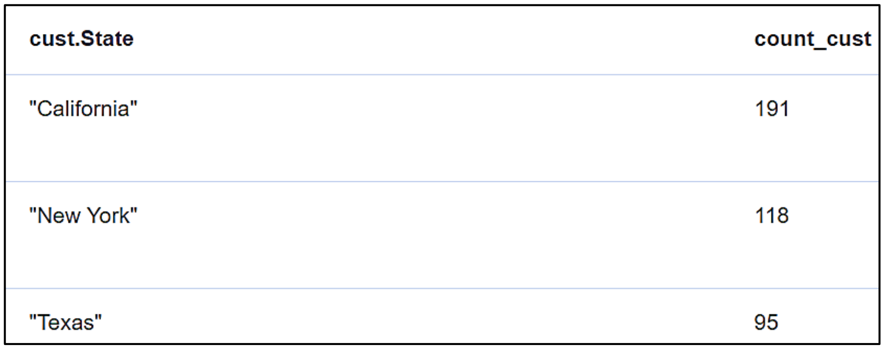
```
MATCH (d:Drone)
RETURN d.Drone_Id,
round(d.Total_Flight_Time,0) as FlightTime
ORDER BY FlightTime DESC
LIMIT 5;
```



d.Drone_Id	FlightTime
"EX1313"	1980.0
"LU1008"	1960.0
"HP1673"	1960.0

# Query 2: Find the number of customers in each State

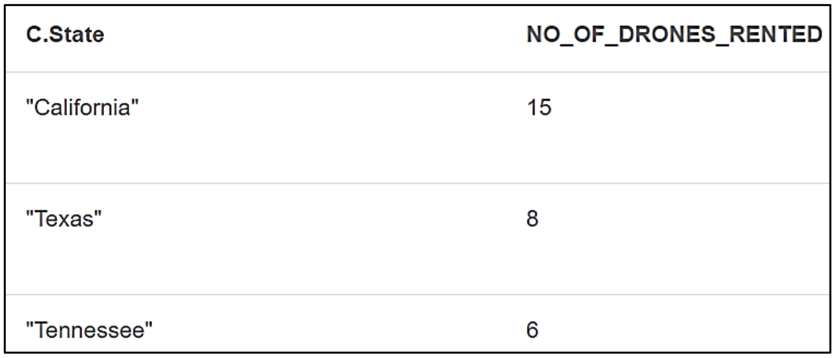
```
MATCH (cust:Customer)
RETURN cust.State ,
count(cust.Customer_Id) as count_cust
ORDER BY count_cust DESC;
```



cust.State	count_cust
"California"	191
"New York"	118
"Texas"	95

# Query 3: Find out which State had the largest number of drones rented in 2020

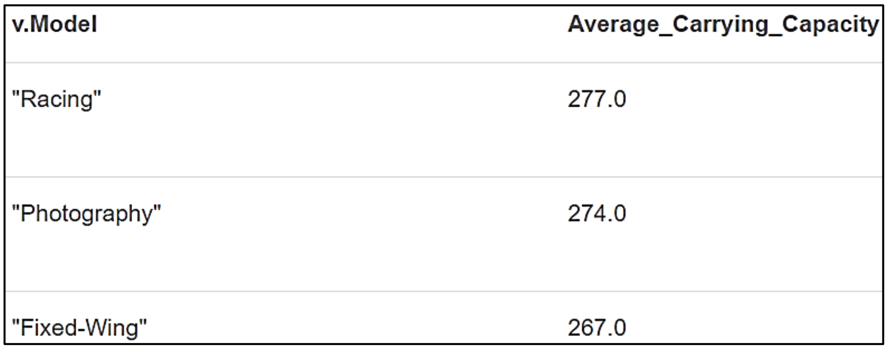
```
MATCH (R:Rents)-[:Rented_By]-(C:Customer)
WHERE (R.Given_time.year) = 2020
RETURN C.State, COUNT(R.Rental_No)
as NO_OF_DRONES_RENTED
ORDER BY NO_OF_DRONES_RENTED
DESC;
```



C.State	NO_OF_DRONES_RENTED
"California"	15
"Texas"	8
"Tennessee"	6

# Query 4: Find the Average Carrying Capacity of each Drone model

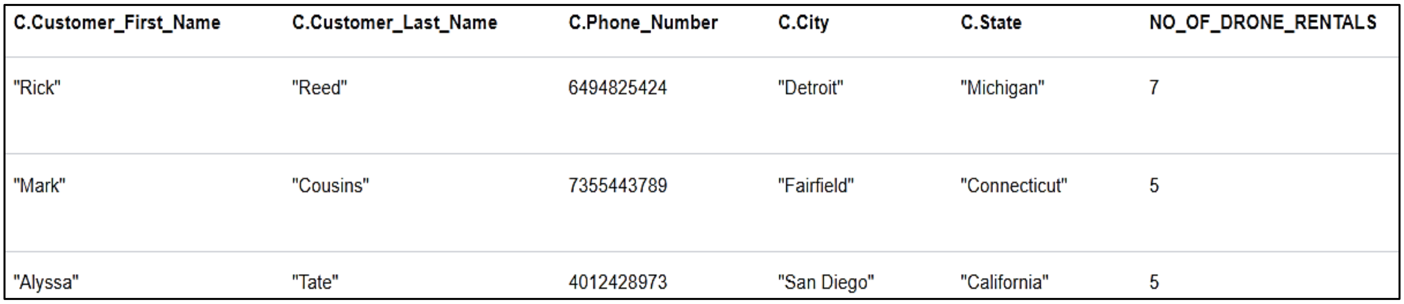
```
 MATCH (c:Company)--(v:Value)
RETURN v.Model,
round(avg(c.Carrying_Capacity),0)
as Average_Carrying_Capacity
ORDER BY Average_Carrying_Capacity
DESC;
```



v.Model	Average_Carrying_Capacity
"Racing"	277.0
"Photography"	274.0
"Fixed-Wing"	267.0

# Query 5: Find out the Customer Details of Customers who had highest no of drone rentals and returned them without damage

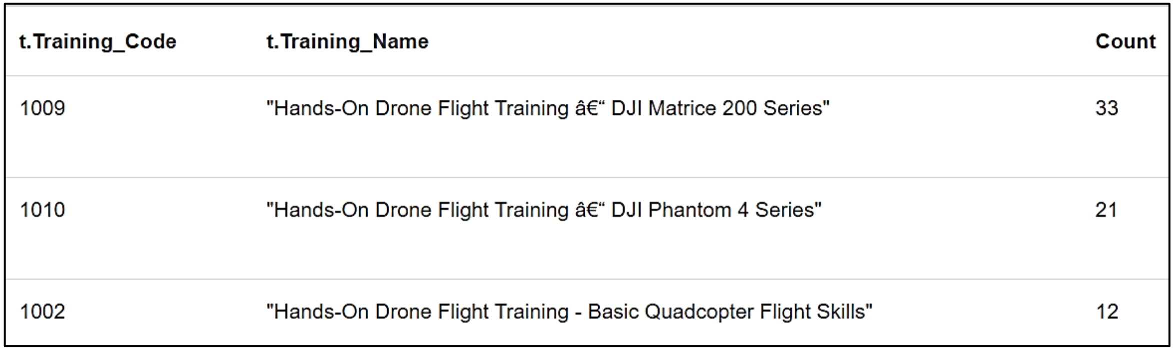
MATCH (R:Rents)-[:Rented_By]->(C:Customer)
WHERE R.Conditions <> 'bad'
RETURN C.Customer_First_Name, C.Customer_Last_Name, C.Phone_Number, C.City, C.State, COUNT(*) AS NO_OF_DRONE_RENTALS
ORDER BY NO_OF_DRONE_RENTALS DESC;



C.Customer_First_Name	C.Customer_Last_Name	C.Phone_Number	C.City	C.State	NO_OF_DRONE_RENTALS
"Rick"	"Reed"	6494825424	"Detroit"	"Michigan"	7
"Mark"	"Cousins"	7355443789	"Fairfield"	"Connecticut"	5
"Alyssa"	"Tate"	4012428973	"San Diego"	"California"	5

# Query 6: Number of times a course was taken up by customers

Match (c:Customer)<-[:Learned_By]-(l:Learns)-[:Training_Of]->(t:TrainingCourse)
return t.Training_Code,
t.Training_Name,
count(c.Customer_Id) as Count
order by Count desc;



t.Training_Code	t.Training_Name	Count
1009	"Hands-On Drone Flight Training â€“ DJI Matrice 200 Series"	33
1010	"Hands-On Drone Flight Training â€“ DJI Phantom 4 Series"	21
1002	"Hands-On Drone Flight Training - Basic Quadcopter Flight Skills"	12

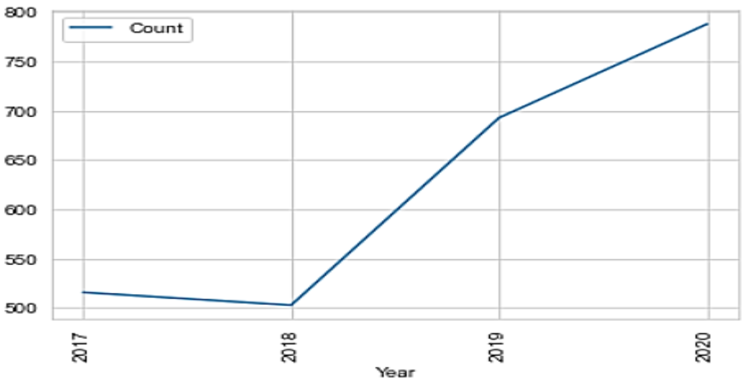
VI. Data retrieval from Database by integrating with Python & Tableau

Python Integration:

MySQL was connected to Python by using **pymysql.connect**(), followed by **cur.excecute()** to execute the SQL query. The results were stored in a dataframe using the **fetchall**() command and **pandas** library. Then, by using **matplotlib** and **seaborn** packages, the graphs were plotted for visual analytics.

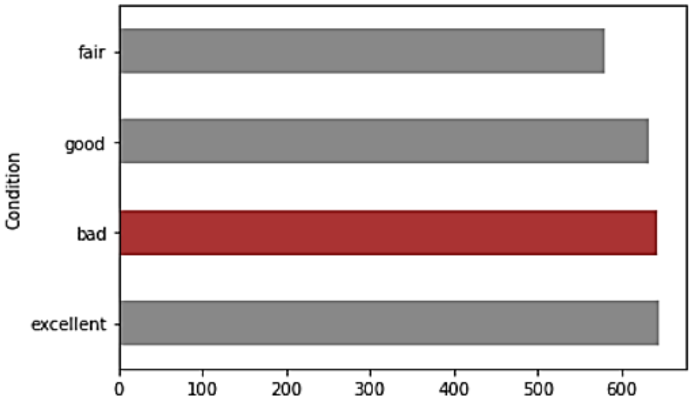
# Query 1: Find the number of drones purchased in the past 5 years

select year(purchase_date), count(Drone_Id)
from Drones
group by year(purchase_date)
order by year(purchase_date);



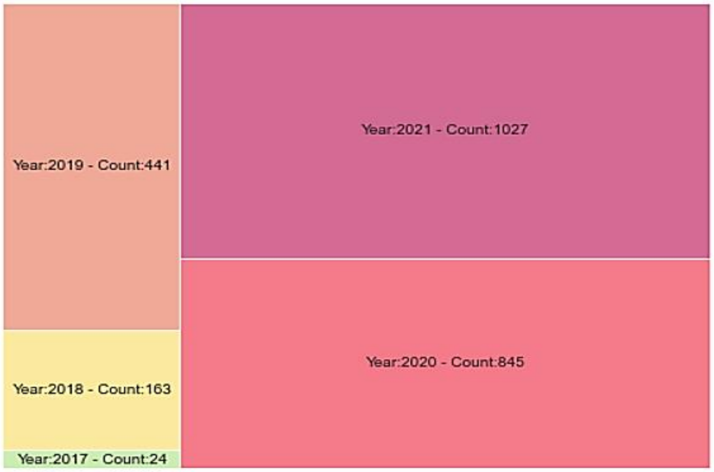
# Query 2: Find the count of drones based on condition upon return and status of rental bond

```
select conditions, rental_bond, count(*) as count
from Rents
group by 1,  2;
```



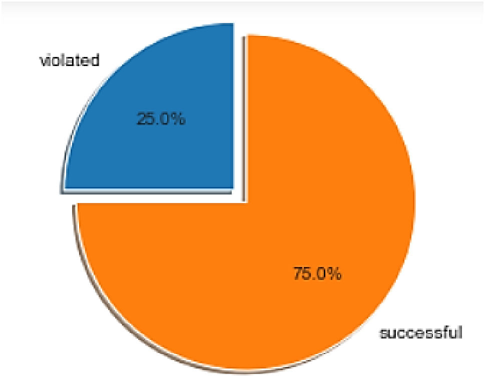
# Query 3: Find the number of drones rented in the past 5 years

```
select year(given_time), count(rental_no)
from Rents
group by year(given_time)
order by year(given_time);
```



# Query 4: Find the percentage of rental bonds violated based on return condition of drones

```
select Rental_Bond,
round((count(rental_no)*100/(select count(*)
                                        from Rents)),0)
                                         as percentage
from Rents
group by (Rental_Bond);
```



# Query 5: Find the count of drones belonging to each Model type

```
select c.Model, count(d.Drone_Id) as Count
from Drones as d, Company as c
where d.Type_Code = c.Type_Code
group by c.Model
order by c.Model;
```

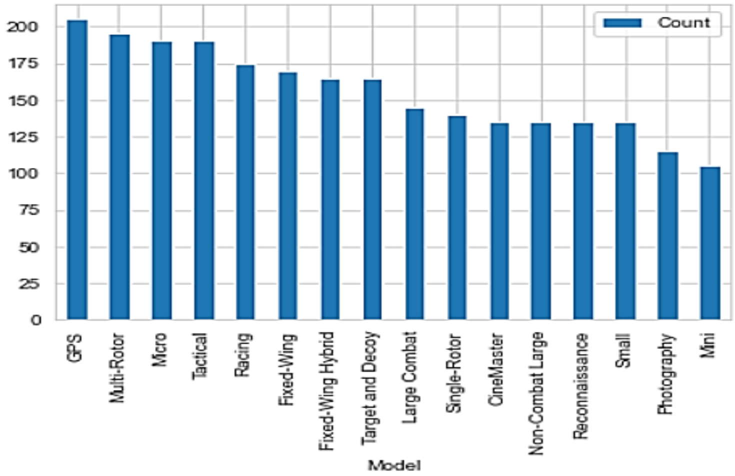
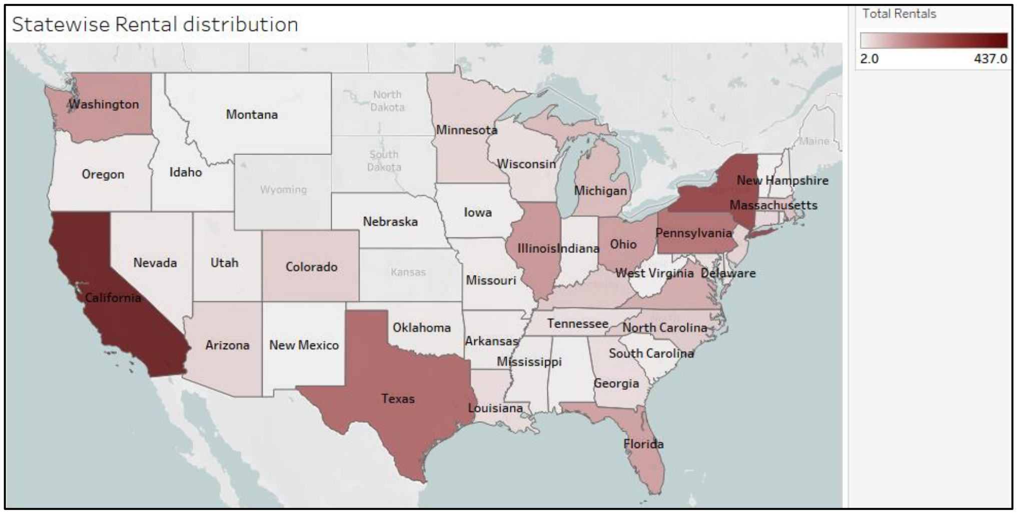


Tableau Integration:

A new connection using the MySQL localhost was setup in Tableau for accessing the Database tables. Thereafter, in the Tableau data source, custom SQL queries were created such that Tableau could interpret the requirement intuitively and as a result, the required visualizations could be generated.

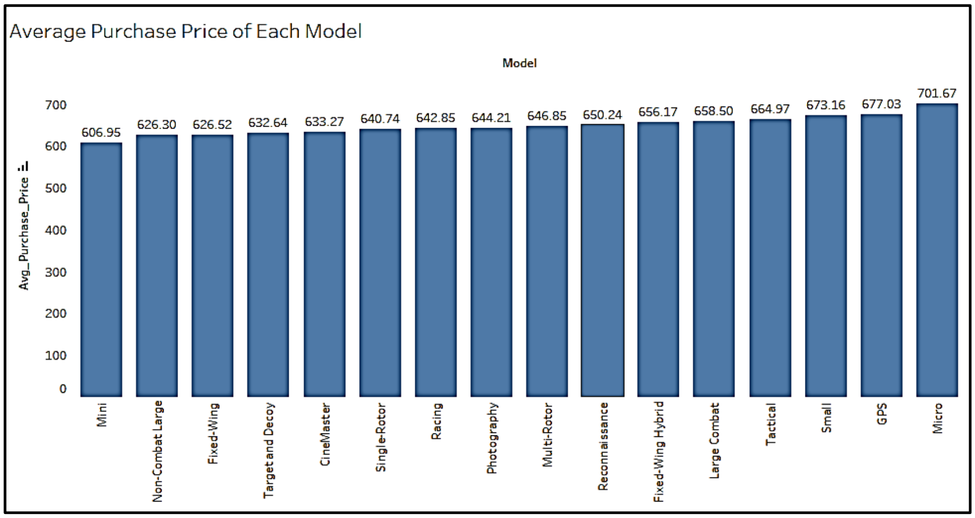
# Query 1: Find the number of drones rented in each State

```
select `customer`.`state` as `state`,
sum(1) as `total rentals`
from `customer`inner join `rents`
on (`customer`.`customer_id` =
`rents`.`customer_id`)
group by 1
```



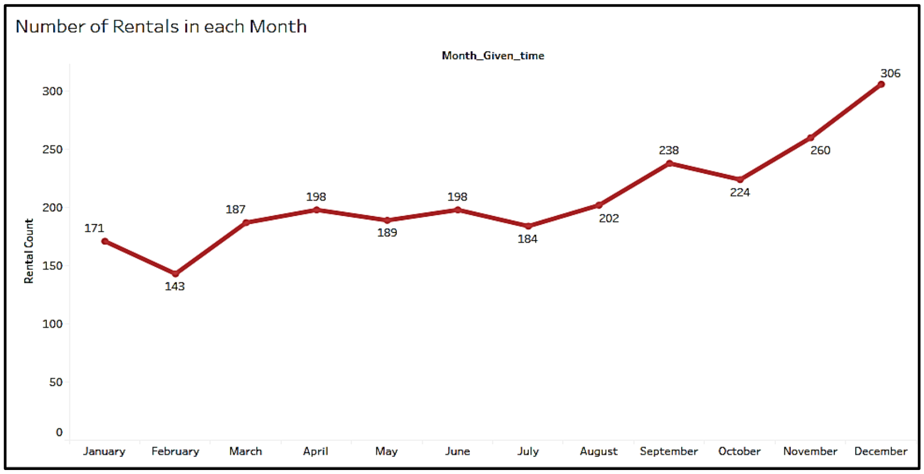
# Query 2: Find the average price for each drone model

```
select `company`.`model` as `model`,
avg(`drones`.`purchase_price`) as
`avg_purchase_price_ok`
from `company`
inner join `drones`
on (`company`.`type_code` =
`drones`.`type_code`)
group by 1
```



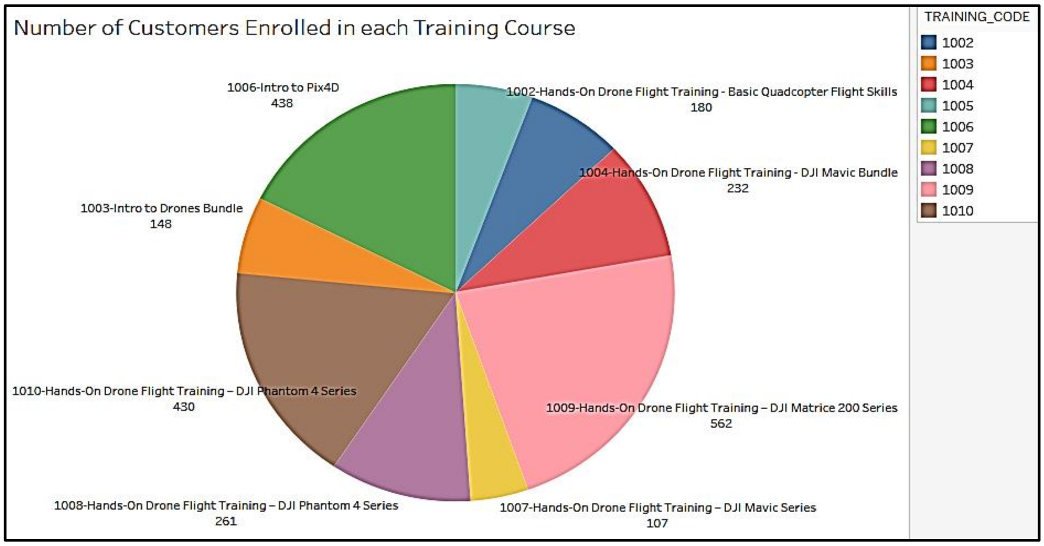
# Query 3: Find the number of drones rented per month

```
select sum(1) as `rental count`,
monthname(`rents`.`given_time`)
as `month_given_time`
from `rents`
group by 2
```



# Query 4: Find the number of Customers enrolled in each training course

```
select `learns`.`training_code` as `training_code`, `custom sql query`.`training_name` as `training_name`,
sum(1) as `customer count`
from `learns` inner join (select `training_course`.`training_code` as `training_code (training_course)`,
                                         `training_course`.`training_name` as `training_name`,
                                        `training_course`.`no_of_hours` as `no_of_hours`,
                                        `training_course`.`training_description` as `training_description`
                                        from `training_course`) `custom sql query`
on (`learns`.`training_code` = `custom sql query`.`training_code (training_course)`)
group by 1, 2
```



VII. Summary and Recommendation

The database created using MySQL for the **HiFlying Drones** company would help a great deal in efficiently maintaining and fetching records which are consistent and less prone to error. The database would ensure better security and information hiding by use of views and access authorization. Eventually, the increased data integrity will lead to independence from other application programs and facilitate the development of front-end and back-end systems for the company. The enhancement of creating an online portal for their users would increase their revenue, and in turn help them make informed business decisions through information engineering and analytics. As a recommendation, the company should take appropriate data governance and storage measures to make sure that high data quality and consistent backups are maintained at all times.