****Project: Clybourn Real Estate Consulting Firm - Database Application****

**Scope and description of project**

The Clybourn Real Estate consulting firm is a firm that helps its clients buy and rent properties. An owner sells their properties through the real estate company. The company maintains a list of all real estate properties that need to be sold or leased. A property will have an owner associated with it. The companies’ agents manage the entire consulting process.

First a client will have to be registered to consulting firm in order to receive consulting services. After registration, for every new request from client to buy/rent a property, an order will be created with a particular agent assigned to that client. An agent will have one or more orders associated with it to work on, similarly a client can place one or more order(s). With reference to an order, agent will then visit properties with client. A property may be visited by many agents and clients. After subsequent visits, final deal will be done.

Each agent will have a rate that would determine its share in case he/she has helped the clients to get the property of their choice. Share of agent depends on the price of deal done i.e. if agent has helped client in deal of property whose final price is $50,000 and agent rate is 10% then agent will get 10% of $50,000 which is $5000. For each order, only one final deal on one particular property will be done. The company also encourages its clients to give rating to agents depending on their services. This parameter helps the consulting firm to determine performance and efficiency of agents.

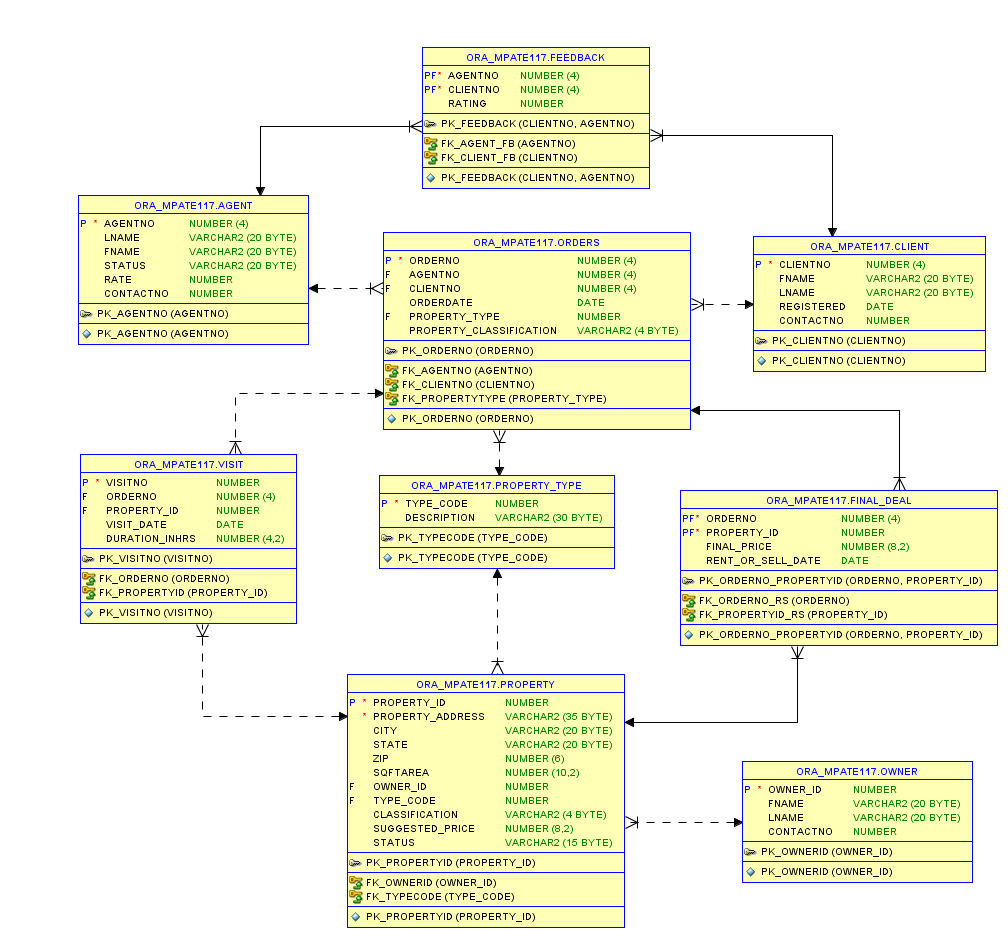
Approach to solution plan

The database designed for the above mentioned real estate consulting firm will contain nine tables. List of all the tables, its purposes and description is described below.

1. **Agent –** contains information of various agents working for consulting firm including rate of each agent.
2. **Client –** contains information of various clients who have registered with consulting firm.
3. **Property\_type –** lists different kinds of properties like apartment, warehouse, shops, offices, vacant land etc.
4. **Owner –** contains basic information about property owners who need to be contacted in case any client is interested in buying/leasing their property.
5. **Property –** list of all properties with address, city, state, square feet area, suggested price that needs to be sold or leased. A property will belong to a property type.
6. **Orders –** encompasses details of orders placed by client with information such as what kind of property client is looking for, order date and agent number. Whenever an order is placed, an agent will be assigned for that order.An order would be associated to one agent and client however one agent can have many orders. Similarly a client can place one or more orders.
7. **Visit –** contains information of various visits done with respect to an order (which would involve agent and client) on different properties. An order would have one or many visits to properties. Similarly a property can be visited by one or more agents and clients.
8. **Final\_deal –** consist of information related to final\_deal like final price, rent or sold date and who were involved in that deal (agent, client and property). For one order, only one final deal would take place. A property can be in one or more final deal only if it is listed for lease. In other words, a property which is listed for selling will have only one final deal associated with it and a property which is listed for lease may have one or more final deal associated with it
9. **Feedback –** stores the rating that client have given to agents depending on services offered by agents. This table would contain unique combination of agentno and clientno as primary key. This means one client cannot give rating more than once to same agent.

E-R Diagram

DBMS considered for this project is Oracle 12c. Below is the E-R Diagram generated through Oracle SQL Data modeler.



# **Create table and Insert table queries**

* CREATE table queries

CREATE TABLE property\_type

(type\_code number,

description varchar2(30),

CONSTRAINT pk\_typecode PRIMARY KEY(type\_code));

CREATE TABLE owner

(owner\_id number,

fname varchar2(20),

lname varchar2(20),

contactno number,

CONSTRAINT pk\_ownerid PRIMARY KEY(owner\_id));

CREATE TABLE property

(property\_id number,

property\_address varchar2(35) NOT NULL,

city varchar2(20),

state varchar2(20),

zip number(6),

sqftarea number(10,2),

owner\_id number,

type\_code number,

classification varchar2(4),

suggested\_price number(8,2),

status varchar2(15),

CONSTRAINT pk\_propertyid PRIMARY KEY(property\_id),

CONSTRAINT fk\_ownerid FOREIGN KEY(owner\_id) REFERENCES owner(owner\_id),

CONSTRAINT fk\_typecode FOREIGN KEY(type\_code) REFERENCES property\_type(type\_code),

CONSTRAINT chk\_classify CHECK (classification IN ('RENT','SALE')),

CONSTRAINT chk\_status CHECK (status IN ('AVAILABLE','RENTED','SOLD')));

CREATE TABLE agent

(agentno number(4),

lname varchar2(20),

fname varchar2(20),

status varchar2(20),

rate number,

contactno number,

CONSTRAINT pk\_agentno PRIMARY KEY (agentno));

CREATE TABLE client

(clientno number(4),

fname varchar2(20),

lname varchar2(20),

registered date,

contactno number,

CONSTRAINT pk\_clientno PRIMARY KEY (clientno));

CREATE TABLE orders

(orderno number(4),

agentno number(4),

clientno number(4),

orderdate date,

property\_type number,

property\_classification varchar2(4),

CONSTRAINT pk\_orderno PRIMARY KEY (orderno),

CONSTRAINT fk\_agentno FOREIGN KEY(agentno) REFERENCES agent(agentno),

CONSTRAINT fk\_clientno FOREIGN KEY(clientno) REFERENCES client(clientno),

CONSTRAINT fk\_propertytype FOREIGN KEY(property\_type) REFERENCES property\_type(type\_code),

CONSTRAINT chk\_proclassify CHECK (property\_classification IN ('RENT','BUY')));

CREATE TABLE visit

(visitno number,

orderno number(4),

property\_id number,

visit\_date date,

duration\_inhrs number(4,2),

CONSTRAINT pk\_visitno PRIMARY KEY (visitno),

CONSTRAINT fk\_orderno FOREIGN KEY(orderno) REFERENCES orders(orderno),

CONSTRAINT fk\_propertyid FOREIGN KEY(property\_id) REFERENCES property(property\_id));

CREATE TABLE final\_deal

(orderno number,

property\_id number,

final\_price number(8,2),

rent\_or\_sell\_date date,

CONSTRAINT pk\_orderno\_propertyid PRIMARY KEY(orderno,property\_id),

CONSTRAINT fk\_orderno\_rs FOREIGN KEY(orderno) REFERENCES orders(orderno),

CONSTRAINT fk\_propertyid\_rs FOREIGN KEY(property\_id) REFERENCES property(property\_id));

CREATE TABLE feedback

(

agentno number(4),

clientno number(4),

rating number,

CONSTRAINT pk\_feedback PRIMARY KEY(clientno,agentno),

CONSTRAINT fk\_agent\_fb FOREIGN KEY(agentno) REFERENCES agent(agentno),

CONSTRAINT fk\_client\_fb FOREIGN KEY(clientno) REFERENCES client(clientno));

* INSERT INTO TABLE queries & Snapshots of table records

INSERT INTO property\_type VALUES(1,'Apartment');

INSERT INTO property\_type VALUES(2,'Single-family house');

INSERT INTO property\_type VALUES(3,'Multi-family house');

INSERT INTO property\_type VALUES(4,'Town house/Row house');

INSERT INTO property\_type VALUES(5,'Cooperative');

INSERT INTO property\_type VALUES(6,'Shop');

INSERT INTO property\_type VALUES(7,'Retail stores');

INSERT INTO property\_type VALUES(8,'Office building');

INSERT INTO property\_type VALUES(9,'Warehouse');

INSERT INTO property\_type VALUES(10,'Vacant land');

INSERT INTO owner VALUES(1001,'Meghna','Patel',6308152870);

INSERT INTO owner VALUES(1002,'Prarthana','Shah',6308152871);

INSERT INTO owner VALUES(1003,'Shivani','Tamhane',6308152872);

INSERT INTO owner VALUES(1004,'Pratishtha','Verma',6308152873);

INSERT INTO owner VALUES(1005,'Sushma','Mahadevan',6308152874);

INSERT INTO owner VALUES(1006,'Neel','Shah',6308152875);

INSERT INTO owner VALUES(1007,'Ankit','Sankhere',6308152876);

INSERT INTO owner VALUES(1008,'Ankur','Bhayani',6308152877);

INSERT INTO owner VALUES(1009,'Abhi','Purohit',6308152878);

INSERT INTO owner VALUES(1010,'Brijesh','Patel',6308152879);

INSERT INTO property VALUES(5001,'1500 South King Drive','Chicago','IL',60616,840,1005,2,'RENT',3000,'AVAILABLE');

INSERT INTO property VALUES(5002,'2455 Prairie Street','Chicago','IL',60623,800,1005,2,'RENT',3500,'AVAILABLE');

INSERT INTO property VALUES(5003,'1677 Indiana Avenue','Chicago','IL',60678,900,1005,3,'SALE',4000,'AVAILABLE');

INSERT INTO property VALUES(5004,'2901 South King Drive','Chicago','IL',60616,840,1005,1,'RENT',1216,'AVAILABLE');

INSERT INTO property VALUES(5005,'2951 South King Drive','Chicago','IL',60616,900,1003,1,'RENT',1500,'AVAILABLE');

INSERT INTO property VALUES(5006,'3001 South King Drive','Chicago','IL',60616,950,1003,1,'RENT',2000,'AVAILABLE');

INSERT INTO property VALUES(5007,'2851 South King Drive','Chicago','IL',60616,800,1002,1,'RENT',1100,'AVAILABLE');

INSERT INTO property VALUES(5008,'2801 South King Drive','Chicago','IL',60616,750,1001,1,'SALE',3000,'AVAILABLE');

INSERT INTO property VALUES(5009,'5555 Carol Stream','Chicago','IL',60567,1200,1010,2,'SALE',5000,'AVAILABLE');

INSERT INTO property VALUES(5010,'1700 Madison Street','Chicago','IL',60987,820,1005,6,'RENT',4000,'AVAILABLE');

INSERT INTO property VALUES(5011,'1745 Madison Street','Chicago','IL',60987,800,1001,7,'RENT',4500,'AVAILABLE');

INSERT INTO property VALUES(5012,'1746 Madison Street','Chicago','IL',60987,810,1002,5,'SALE',8000,'AVAILABLE');

INSERT INTO property VALUES(5013,'1766 Madison Street','Chicago','IL',60987,920,1003,4,'SALE',4100,'AVAILABLE');

INSERT INTO property VALUES(5014,'1800 Washington Street','Chicago','IL',60766,950,1009,2,'SALE',6000,'AVAILABLE');

INSERT INTO agent VALUES(2001,'Davis','David','Manager',11,3122345001);

INSERT INTO agent VALUES(2002,'Smith','John','Trainee',10,3122345002);

INSERT INTO agent VALUES(2003,'Jackson','Luke','Manager',12,3122345003);

INSERT INTO agent VALUES(2004,'Bond','James','Manager',13,3122345004);

INSERT INTO agent VALUES(2005,'Matthews','Kate','Supervisor',20,3122345005);

INSERT INTO agent VALUES(2006,'Patel','Ishwar','Manager',15,3122345006);

INSERT INTO agent VALUES(2007,'Chan','Jackie','Manager',16,3122345007);

INSERT INTO agent VALUES(2008,'Wang','Michael','Trainee',8,3122345008);

INSERT INTO agent VALUES(2009,'Adams','Sam','Manager',12,3122345009);

INSERT INTO agent VALUES(2010,'Lee','Bret','Supervisor',11,3122345010);

INSERT INTO client VALUES(3001,'Karishma','Kapoor','20-Feb-15',7713122920);

INSERT INTO client VALUES(3002,'Kareena','Kapoor','23-Feb-15',7713122921);

INSERT INTO client VALUES(3003,'Aishwarya','Rai','25-Feb-15',7713122922);

INSERT INTO client VALUES(3004,'Shahrukh','Khan','01-Mar-15',7713122923);

INSERT INTO client VALUES(3005,'Dilip','Kumar','15-Mar-15',7713122924);

INSERT INTO client VALUES(3006,'Akshay','Kumar','30-Mar-15',7713122925);

INSERT INTO client VALUES(3007,'Aamir','Khan','10-Apr-15',7713122926);

INSERT INTO client VALUES(3008,'Priyanka','Chopra','13-Apr-15',7713122927);

INSERT INTO client VALUES(3009,'Juhi','Chawla','07-May-15',7713122928);

INSERT INTO client VALUES(3010,'Anupam','Kher','27-May-15',7713122929);

INSERT INTO client VALUES(3011,'Preeti','Zinta','30-May-15',7713122930);

INSERT INTO orders VALUES(4001,2004,3003,'01-Mar-2015',1,'RENT');

INSERT INTO orders VALUES(4002,2004,3004,'03-Mar-2015',2,'BUY');

INSERT INTO orders VALUES(4003,2004,3002,'25-Feb-2015',6,'RENT');

INSERT INTO orders VALUES(4004,2005,3001,'22-Feb-2015',3,'BUY');

INSERT INTO orders VALUES(4005,2005,3005,'17-Mar-2015',2,'BUY');

INSERT INTO orders VALUES(4006,2005,3007,'13-Apr-2015',2,'RENT');

INSERT INTO orders VALUES(4007,2007,3006,'02-Apr-2015',7,'RENT');

INSERT INTO orders VALUES(4008,2007,3008,'15-Apr-2015',5,'BUY');

INSERT INTO orders VALUES(4009,2007,3010,'28-May-2015',4,'BUY');

INSERT INTO orders VALUES(4010,2003,3009,'08-May-2015',1,'RENT');

INSERT INTO feedback VALUES(2004,3003,3);

INSERT INTO feedback VALUES(2004,3004,3.5);

INSERT INTO feedback VALUES(2004,3002,4);

INSERT INTO feedback VALUES(2005,3001,3);

INSERT INTO feedback VALUES(2005,3005,5);

INSERT INTO feedback VALUES(2005,3007,4);

INSERT INTO feedback VALUES(2007,3006,3);

INSERT INTO feedback VALUES(2007,3008,3);

INSERT INTO feedback VALUES(2007,3010,2.5);

INSERT INTO feedback VALUES(2003,3009,2);

INSERT INTO visit VALUES(6001,4001,5005,'02-Mar-2015',2);

INSERT INTO visit VALUES(6002,4001,5004,'03-Mar-2015',1.25);

INSERT INTO visit VALUES(6003,4001,5006,'04-Mar-2015',1.5);

INSERT INTO visit VALUES(6004,4001,5007,'05-Mar-2015',1);

INSERT INTO visit VALUES(6005,4002,5009,'03-Mar-2015',2.25);

INSERT INTO visit VALUES(6006,4005,5009,'18-Mar-2015',2.25);

INSERT INTO visit VALUES(6007,4006,5001,'13-Apr-2015',1);

INSERT INTO visit VALUES(6008,4006,5002,'14-Apr-2015',0.75);

INSERT INTO visit VALUES(6009,4010,5005,'14-May-2015',1);

INSERT INTO visit VALUES(6010,4010,5006,'15-May-2015',1.25);

INSERT INTO visit VALUES(6011,4010,5007,'16-May-2015',1.75);

INSERT INTO visit VALUES(6012,4003,5010,'26-Feb-2015',2);

INSERT INTO visit VALUES(6013,4004,5003,'28-Feb-2015',2.75);

INSERT INTO visit VALUES(6014,4007,5011,'02-Apr-2015',1.75);

INSERT INTO visit VALUES(6015,4008,5012,'28-Apr-2015',1.25);

INSERT INTO visit VALUES(6016,4009,5013,'29-May-2015',0.5);

INSERT INTO visit VALUES(6017,4002,5014,'30-May-2015',0.5);

INSERT INTO final\_deal VALUES(4005,5009,4900,'20-Mar-2015');

INSERT INTO final\_deal VALUES(4010,5007,1100,'16-May-2015');

INSERT INTO final\_deal VALUES(4006,5001,3000,'14-Apr-2015');

INSERT INTO final\_deal VALUES(4001,5004,1216,'06-Mar-2015');

INSERT INTO final\_deal VALUES(4003,5010,4000,'27-Feb-2015');

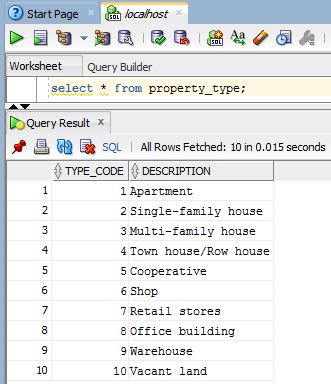
INSERT INTO final\_deal VALUES(4004,5003,4000,'28-Feb-2015');

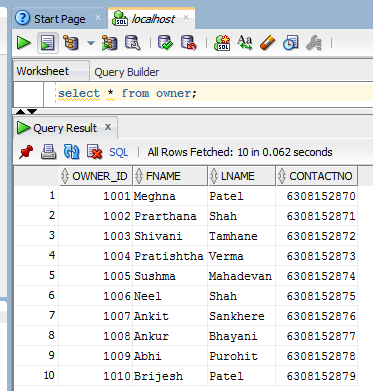
INSERT INTO final\_deal VALUES(4007,5011,4400,'02-Apr-2015');

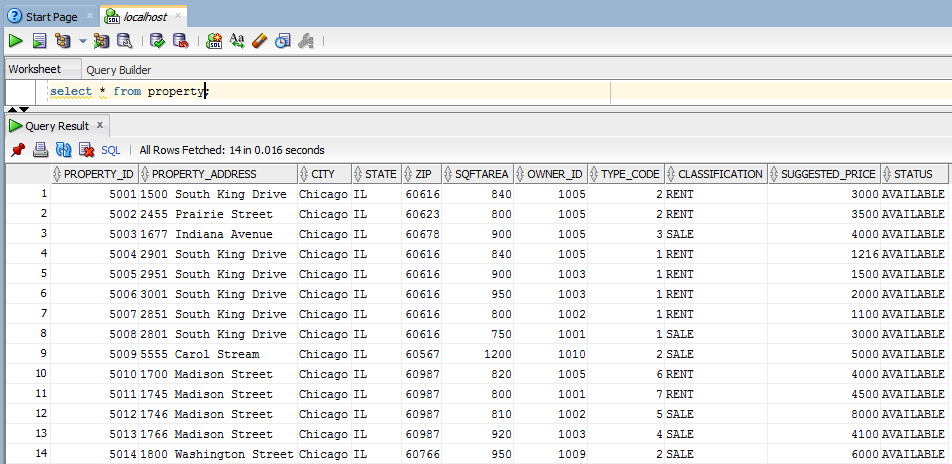
INSERT INTO final\_deal VALUES(4008,5012,8000,'28-Apr-2015');

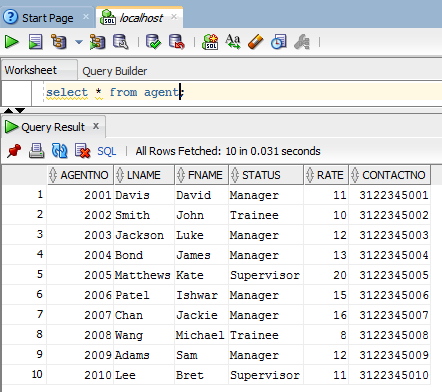
INSERT INTO final\_deal VALUES(4009,5013,4100,'29-May-2015');

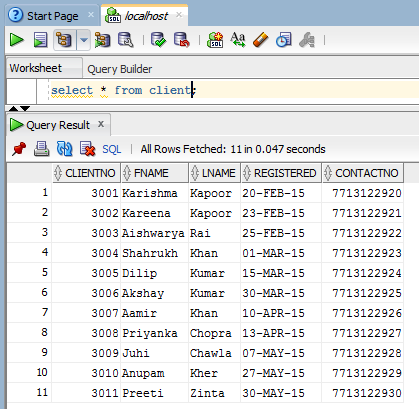
INSERT INTO final\_deal VALUES(4002,5014,6000,'30-May-2015');

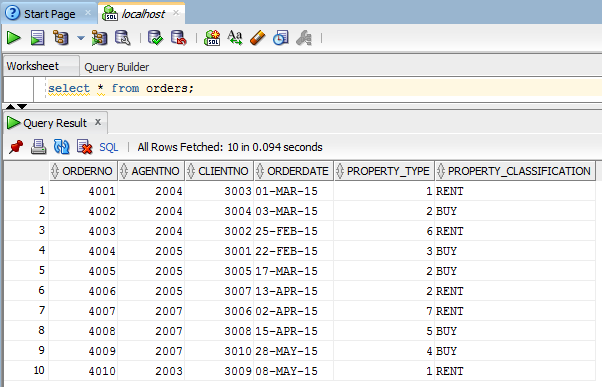


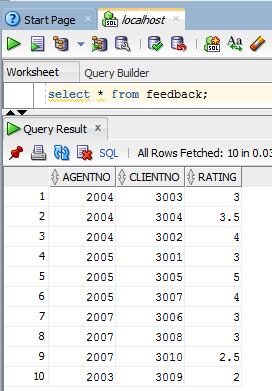


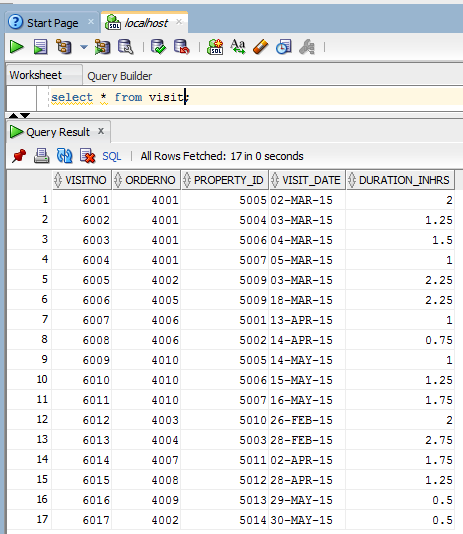


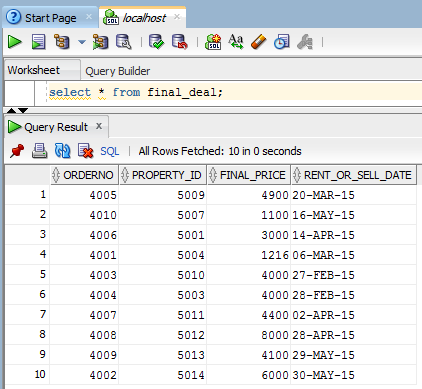












Create Trigger

Depending on the database design, one must ensure that while inserting entries into final\_deal table, the property type and classification of order (the one for which client is looking for) and property type and classification of the property (the one which has been selected for deal) are same.

A trigger has been created to meet this requirement. Below is the code for the same.

set serveroutput on;

CREATE OR REPLACE TRIGGER finaldeal\_trigger

BEFORE INSERT ON FINAL\_DEAL

--REFERENCING OLD as o NEW as n

FOR EACH ROW

DECLARE

o\_ptype ORDERS.PROPERTY\_TYPE%TYPE;

o\_pclass ORDERS.PROPERTY\_CLASSIFICATION%TYPE;

p\_ptype PROPERTY.TYPE\_CODE%TYPE;

p\_pclass PROPERTY.CLASSIFICATION%TYPE;

BEGIN

SELECT property\_type,property\_classification INTO o\_ptype,o\_pclass

FROM orders

WHERE orderno = :NEW.orderno;

SELECT type\_code,classification INTO p\_ptype, p\_pclass

from property

WHERE property\_id = :NEW.property\_id;

IF o\_ptype <> p\_ptype THEN

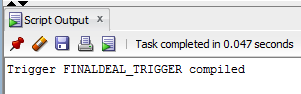
RAISE\_APPLICATION\_ERROR(-20001,'Property type mismatch');

ELSIF o\_pclass <> p\_pclass THEN

RAISE\_APPLICATION\_ERROR(-20002,'Classification mismatch');

END IF;

END;



**Test the trigger**

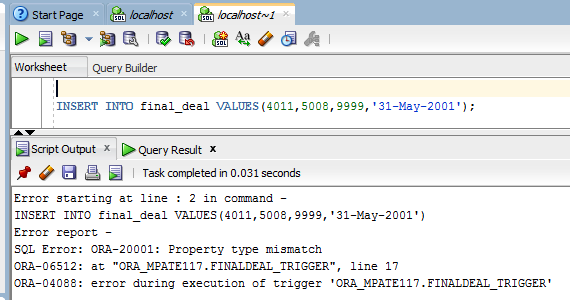
A sample order is created. With reference to this order, try to insert record into final\_deal table where property type or classification differs.

INSERT INTO orders VALUES(4011,2003,3009,'31-May-2001',10,'RENT');

INSERT INTO final\_deal VALUES(4011,5008,9999,'31-May-2001');

Property 5008 has property type 1 and classification SALE. Hence above insert query should raise an exception.

Output:



Perform Analysis of the Data – Advanced SQL queries

1. Average rating of agents

WITH Average AS(select agentno,round(avg(rating),2) as avg

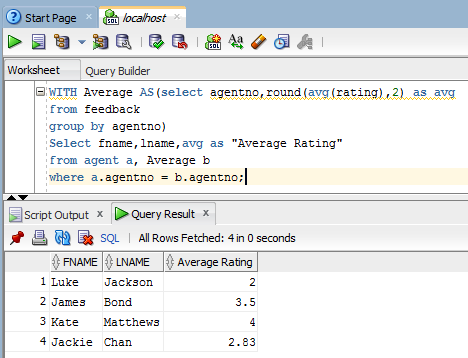
from feedback

group by agentno)

Select fname,lname,avg as "Average Rating"

from agent a, Average b

where a.agentno = b.agentno;



Feedback table stores rating given by client to agents. The above query will perform aggregation and display average rating for each agent who has been rated. This would aid the firm in measuring the performance of agents.

This above mentioned query makes use of WITH clause, which is usually considered for improving performance of queries.

1. Select all agents whose average rating is less than the average rating of all agents in same status (status here is Manager, Supervisor, Trainee etc.)

select ao.agentno,status,trunc(avg(rating),2) AS AGENT\_RATING

from feedback f join agent ao

on f.agentno = ao.agentno

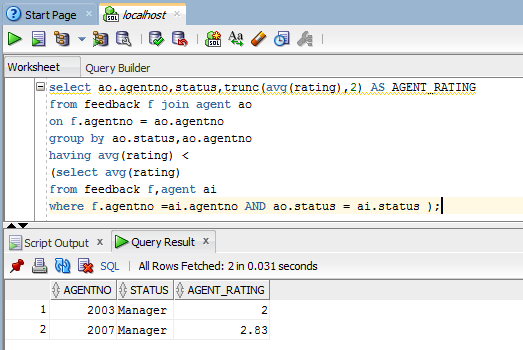
group by ao.status,ao.agentno

having avg(rating) <

(select avg(rating)

from feedback f,agent ai

where f.agentno =ai.agentno AND ao.status = ai.status );



The above mentioned query makes use of co-related sub-query. This analytic can help the firm to know the agents whose performance is below average.

1. List of properties which have not been visited yet.

select property\_id,property\_address,city,sqftarea, description "Type",classification,suggested\_price,status

from property p JOIN property\_type pt

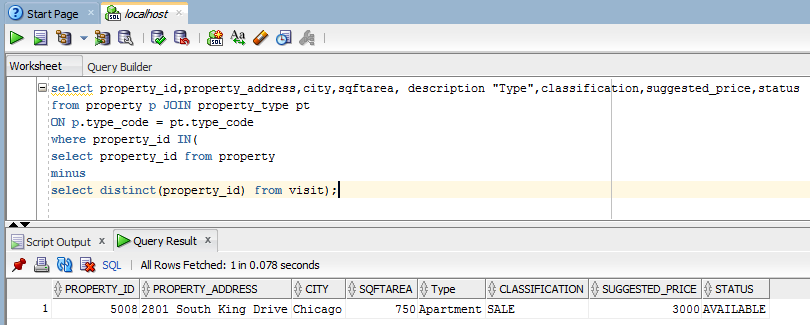
ON p.type\_code = pt.type\_code

where property\_id IN(

select property\_id from property

minus

select distinct(property\_id) from visit);



This query makes use of MINUS operator. Distinct is used to get a list of all unique properties from visit table. Subtracting list of visited properties from list of all the properties would give list of properties which have not been visited.

1. Property type which has highest demand depending on number of orders received

select description, count(\*) as demand

from orders o JOIN property\_type p

on o.property\_type = p.type\_code

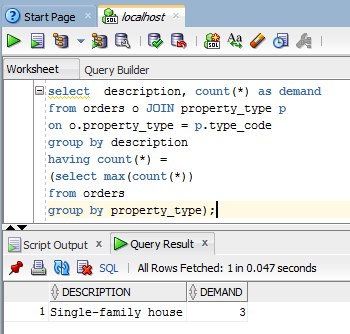
group by description

having count(\*) =

(select max(count(\*))

from orders

group by property\_type);



This query makes use of sub query along with group by and having clause. Knowing the type of property which has highest number of demand can be a useful data analytics.

1. For each agent, find the number of buy and rental orders.

WITH pivot\_data AS

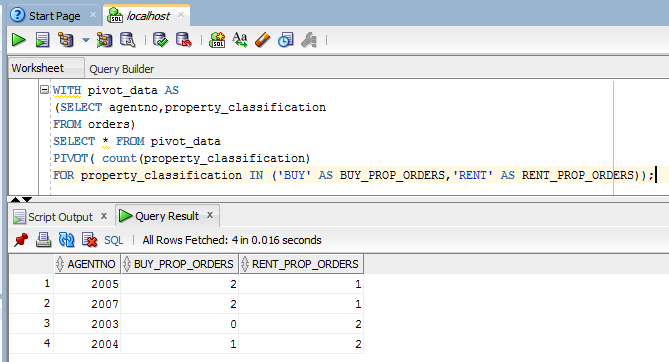
(SELECT agentno,property\_classification

FROM orders)

SELECT \* FROM pivot\_data

PIVOT( count(property\_classification)

FOR property\_classification IN ('BUY' AS BUY\_PROP\_ORDERS,'RENT' AS RENT\_PROP\_ORDERS));



The above mentioned advanced SQL query makes use of PIVOT clause to put rows data values into columns. From this data analytics, the consulting firm can find out the number of sell as well as rental orders taken by various agents till date.

1. Clients who have registered but not yet placed any order.

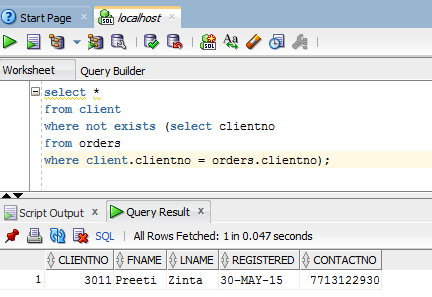
select \*

from client

where not exists (select clientno

from orders

where client.clientno = orders.clientno);



This query makes use of EXISTS operator along with negation NOT operator to find those clients who have registered but not yet placed any order.

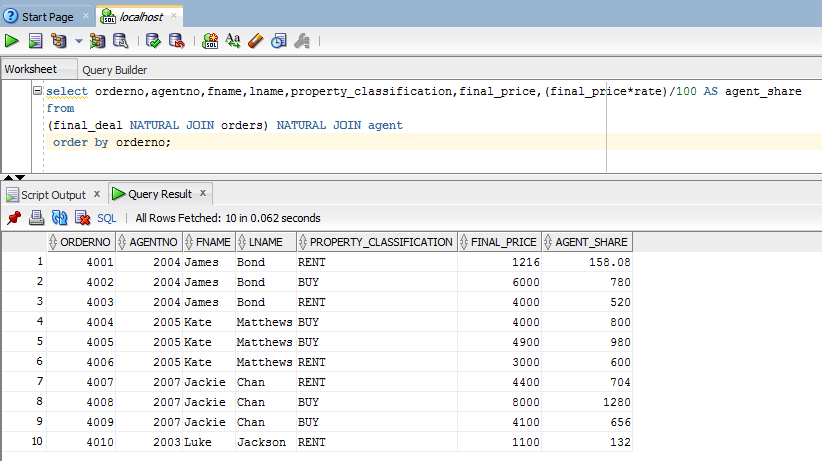
1. For each final deal done, calculate the share of agent depending on agent’s rate and final price of deal done

select orderno,agentno,fname,lname,property\_classification,final\_price,(final\_price\*rate)/100 AS agent\_share

from

(final\_deal NATURAL JOIN orders) NATURAL JOIN agent

order by orderno;



The above mentioned query helps the firm to calculate the agent share for each final deal done. This query uses natural join to join the tables.

1. Top five deals with highest price

select t.orderno, a.fname || ' ' || a.lname as Agent, c.fname || ' ' || c.lname as Client,pt.description AS property\_type,p.classification,t.final\_price,t.rent\_or\_sell\_date

from

(select \* from

(select \* from final\_deal order by final\_price desc)

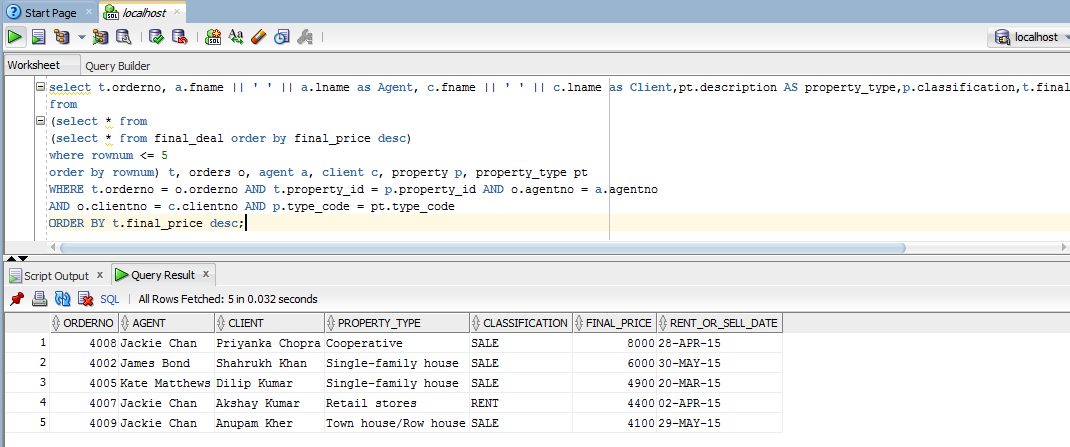
where rownum <= 5

order by rownum) t, orders o, agent a, client c, property p, property\_type pt

WHERE t.orderno = o.orderno AND t.property\_id = p.property\_id AND o.agentno = a.agentno

AND o.clientno = c.clientno AND p.type\_code = pt.type\_code

ORDER BY t.final\_price desc;



This is an example of Top-n analysis using sub queries.

1. List of properties who have been visited more than once

select a.property\_id,b.property\_address,b.city,a.No\_of\_visits

from

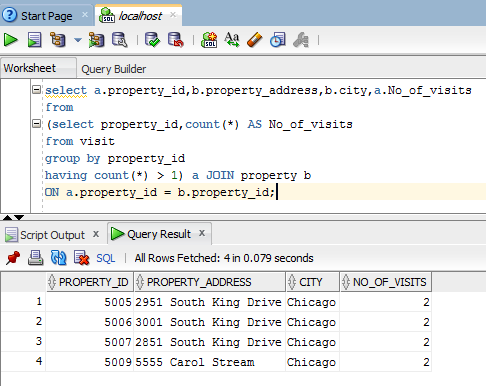
(select property\_id,count(\*) AS No\_of\_visits

from visit

group by property\_id

having count(\*) > 1) a JOIN property b

ON a.property\_id = b.property\_id;



1. Find the number of properties sold and leased by agent

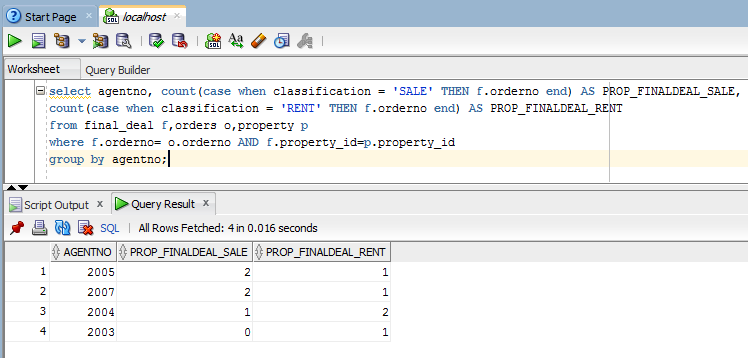
select agentno, count(case when classification = 'SALE' THEN f.orderno end) AS PROP\_FINALDEAL\_SALE,

count(case when classification = 'RENT' THEN f.orderno end) AS PROP\_FINALDEAL\_RENT

from final\_deal f,orders o,property p

where f.orderno= o.orderno AND f.property\_id=p.property\_id

group by agentno;

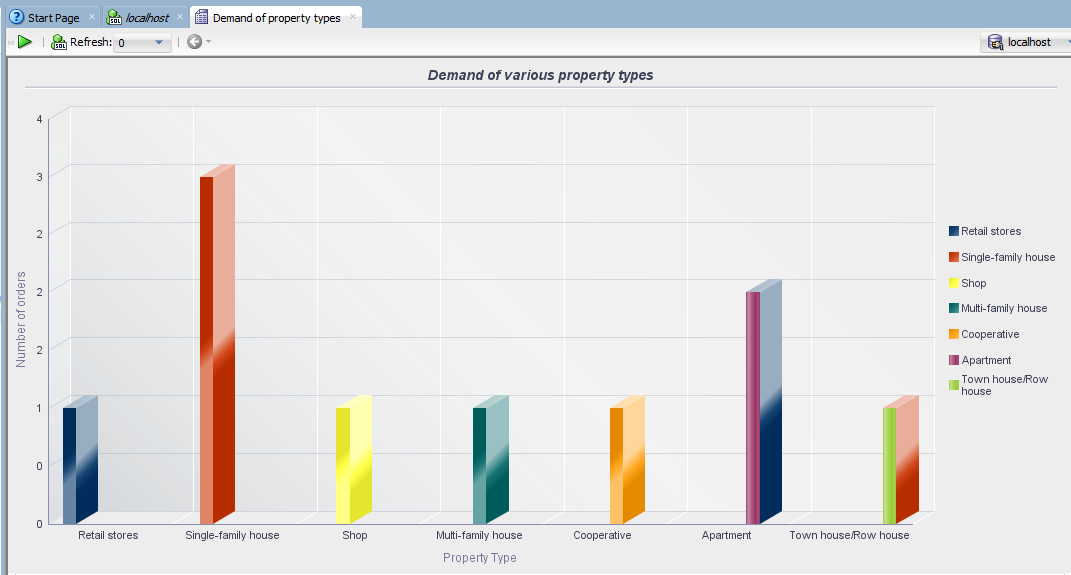


This query makes use of CASE statement within the aggregate function. The output of this query can be a useful data analytic for firm.

Reports, Charts and Graphs

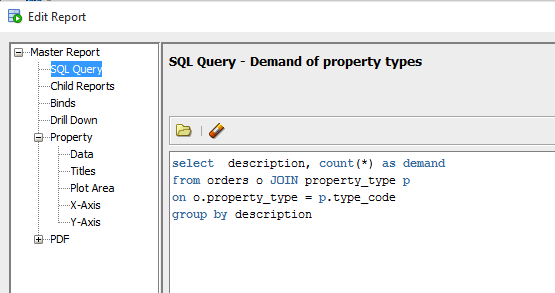
Charts have been prepared in Oracle SQL developer using User-defined reports.

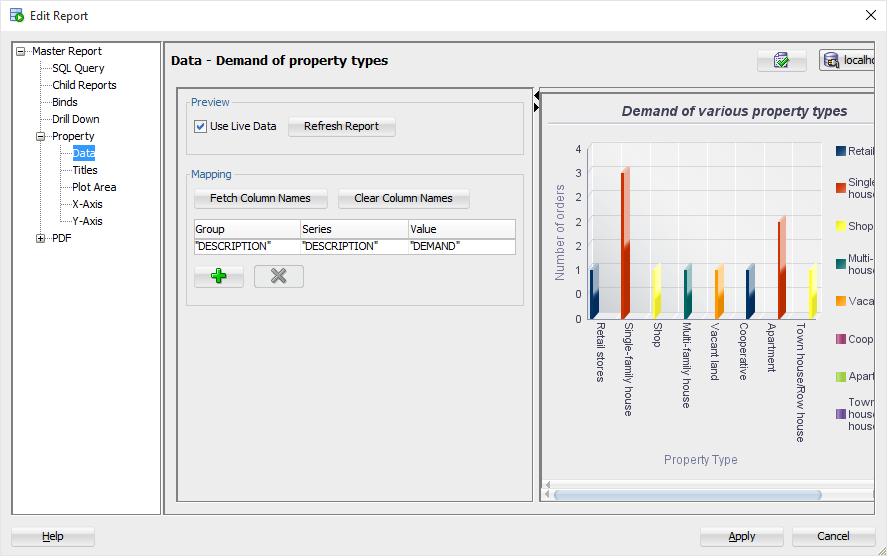
1. Demand of Property types



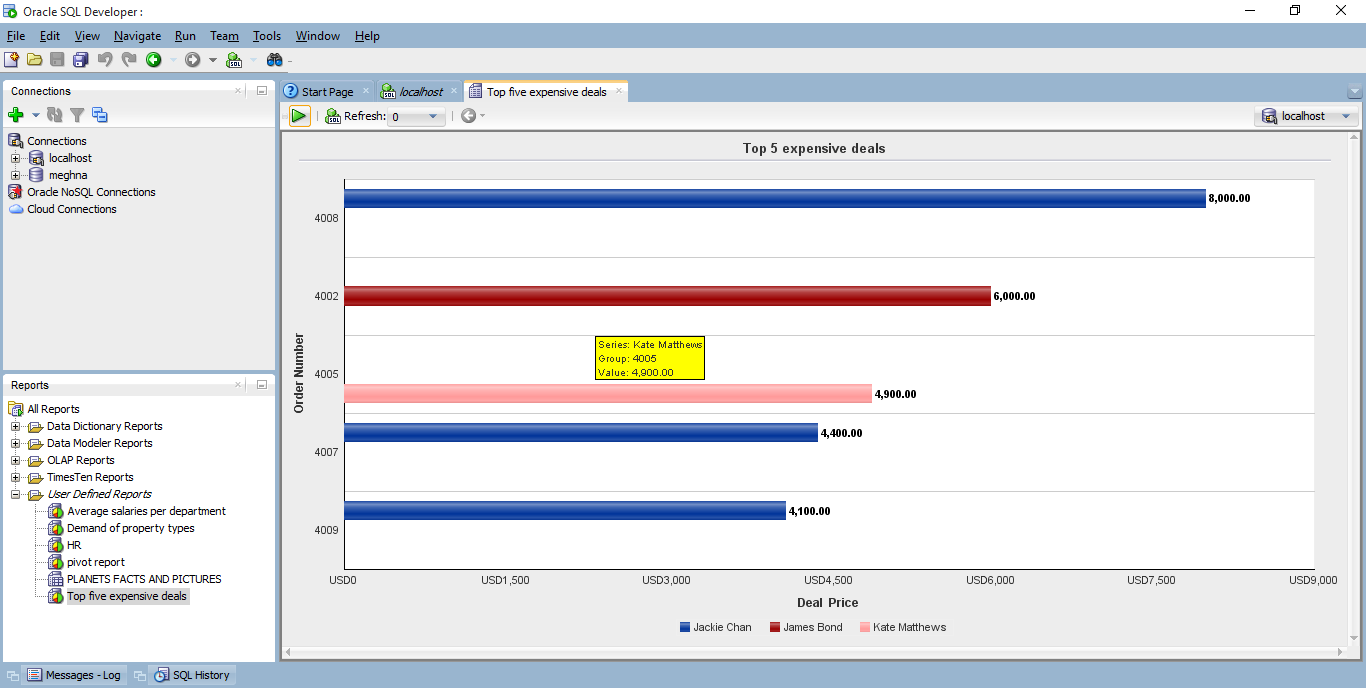
From the chart, it can be inferred that Single-Family house has highest demand, following which Apartment has next highest demand and so on. Demand is calculated on the basis of number of orders received from clients.

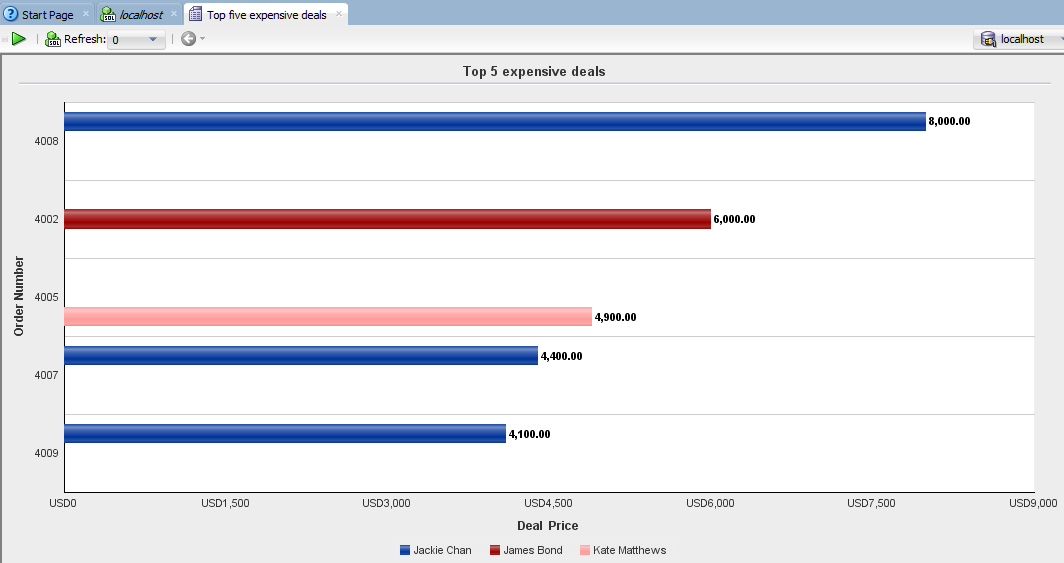
Below is the SQL query for report and other relevant details.





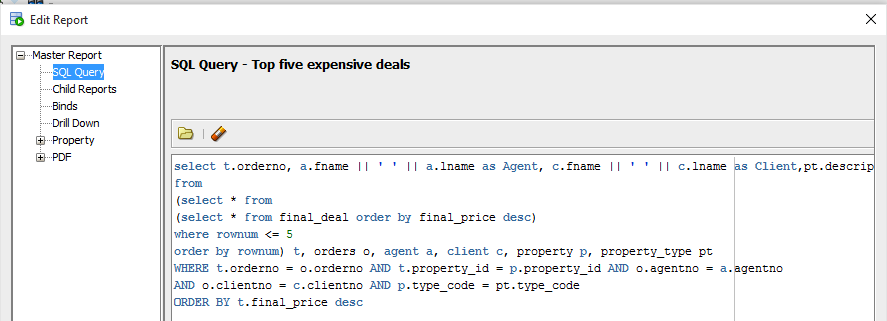
1. Top five expensive deals

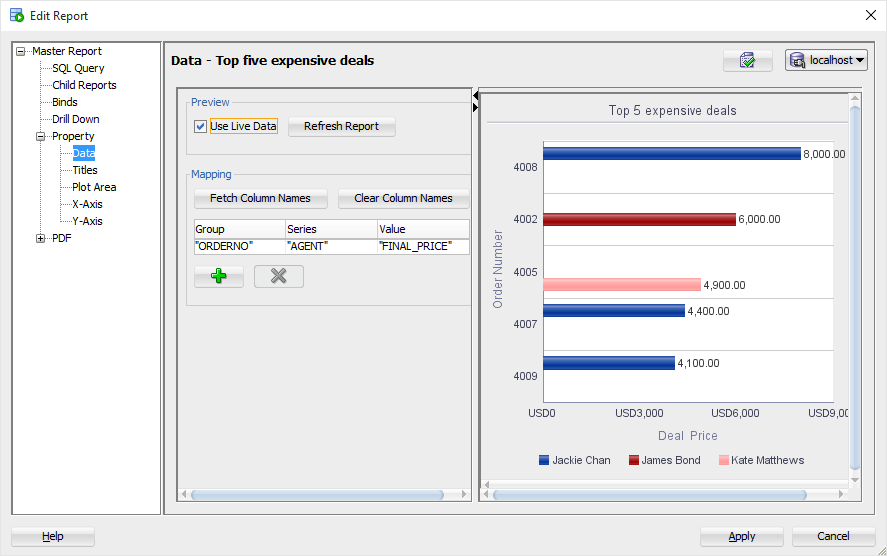




This report displays details of top five expensive deals, which agent is responsible for deal and final price of deal. It can be seen from the graph that out of top five expensive deals, three deals were done by same agent (Jackie Chan). Thus we can say that agent Jackie Chan is most valuable for success of the company.

Below is the SQL query for report and other relevant details.





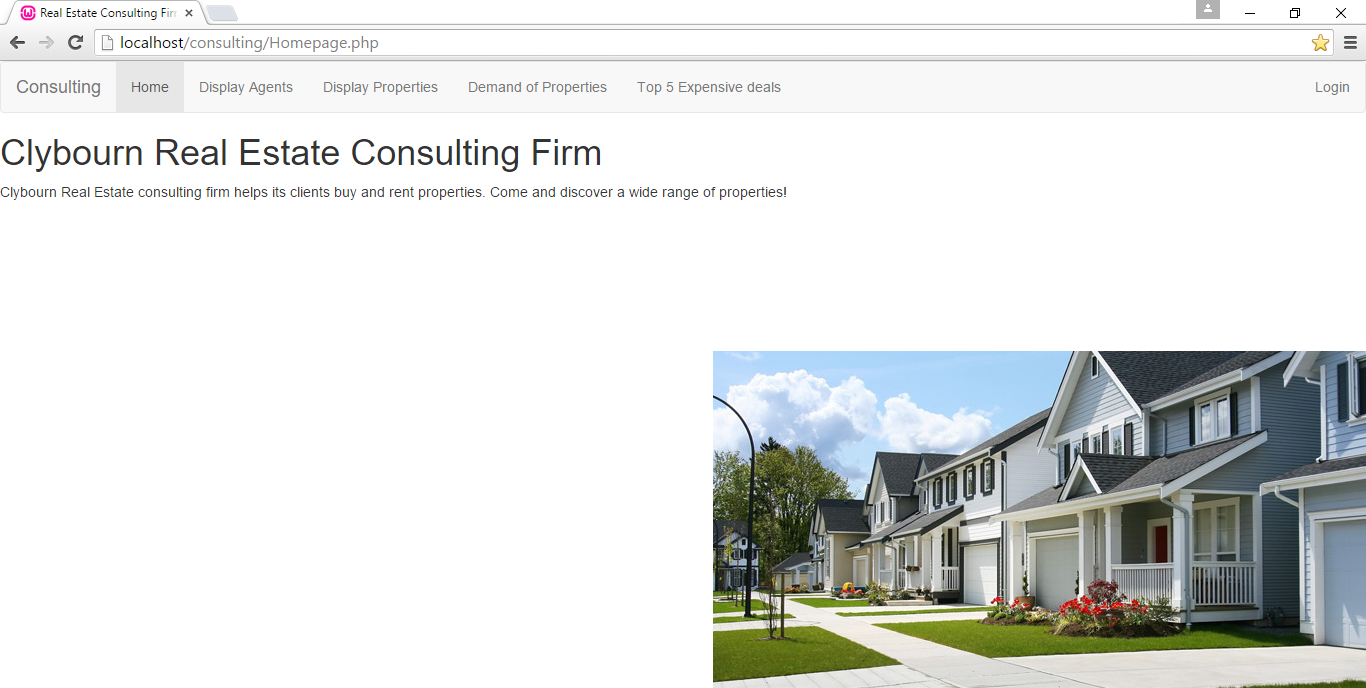
Web database Presence of Project

PHP with Oracle 12c. Data is fetched from localhost instance of database.

Front end: PHP Hypertext Preprocessor

Back end: Oracle 12c

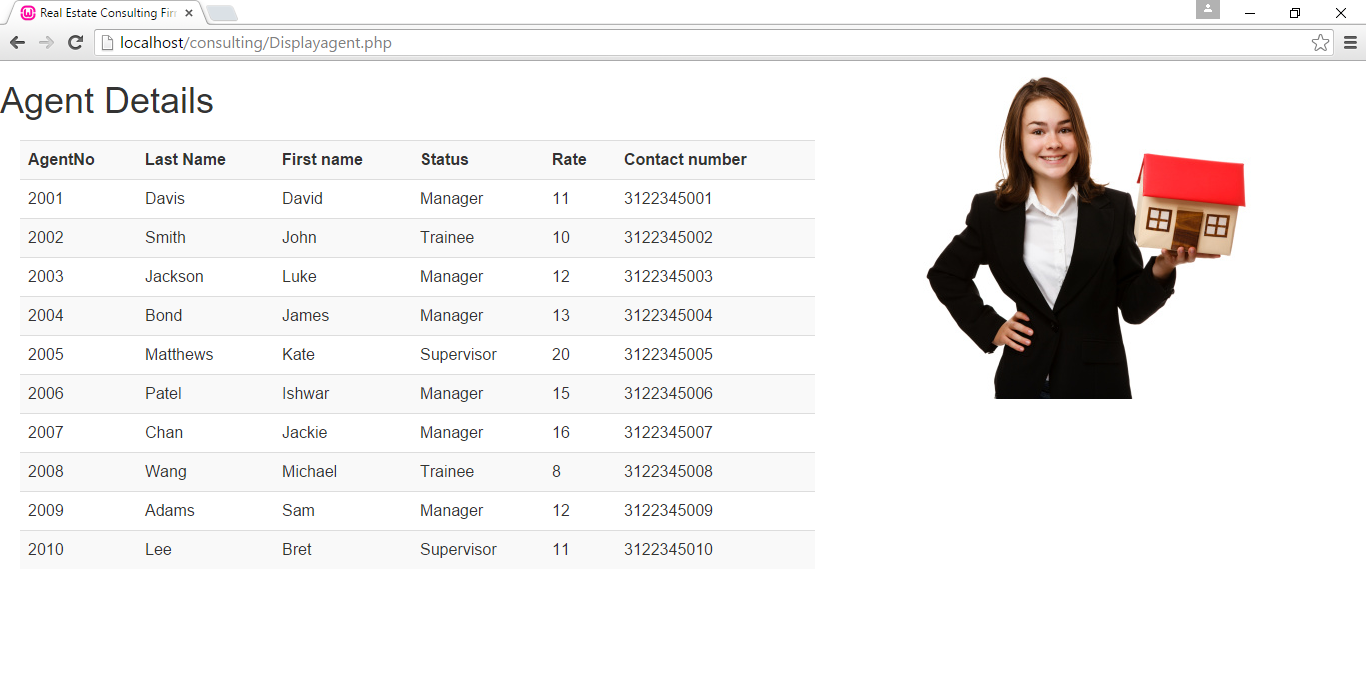
**HomePage.php**



**Displayagent.php**

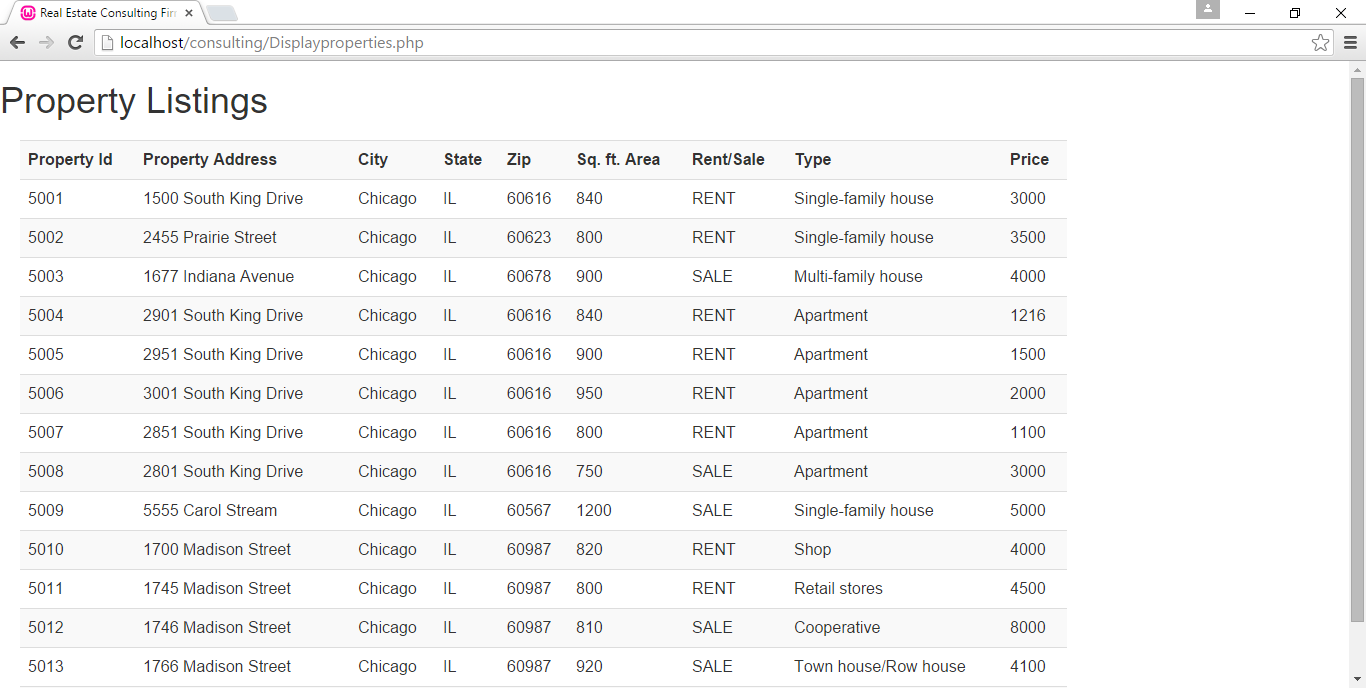
Connection of PHP to oracle database is done using OCI extension. Method oci\_connect() and oci\_close() is used. Data is fetched from agent table after connection to oracle database is established.

This page lists details of all agents who are working for consulting firm.



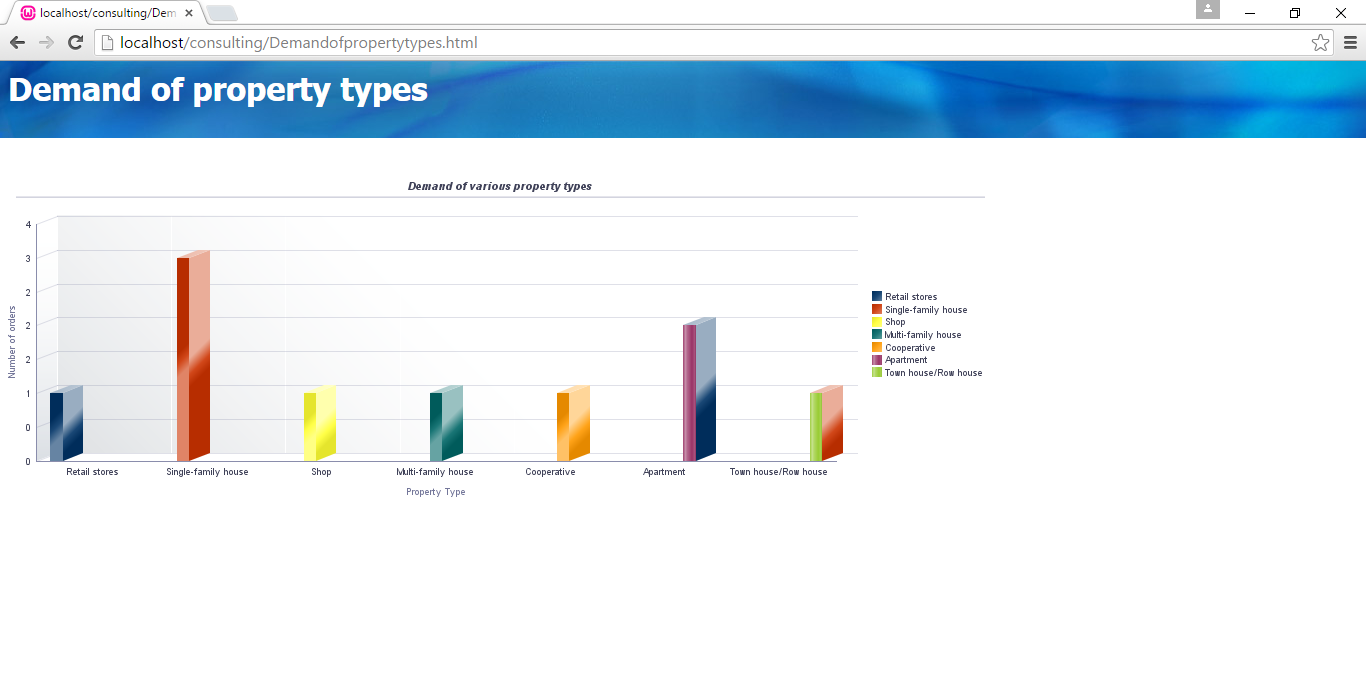
**DisplayProperties.php**

Just as displaying agent, in case of displaying property listings, first connection with oracle database is established and then, data is fetched by joining property and property\_type table and displayed. This page lists all the properties and their necessary information.



**Demandofpropertytypes.html**

The report developed in Oracle developer is saved as .html file and the same is called when user clicks on “Demand of Properties” button on navigation bar.



**Topfiveexpensivedeals.html**

Same as above. Report from Oracle SQL developer is saved as .html file and displayed here.



Maintenance

The project has covered almost all aspects of a Real estate consulting firm. However some enhancements can still be made. Project can be extended to include all the process related to payment to be made to agents. Also scope of this project is confined to a single location. If the consulting firm plans to have many other locations across the country, then accordingly future developments can be done. In addition to this, the firm can also store details of rental contract between client and property owner in case a final deal for leasing the property is done. Doing this, the firm can track when does rental contract of property expires and when the same will be available for lease.

References

<http://www.sreenivaskandakuru.com/2013/10/realestate-agency-data-model.html>

<http://www.albany.edu/~goel/classes/spring2007/pwsp/database/DatabaseDesign.pdf>

<http://lyndiahayden.yolasite.com/resources/317/CIS%20317%20ER%20Diagrams.pdf>

<https://www.rcanalytics.com/glossary/p/property-types.aspx>