UNIVERSITY OF SUNDERLAND
ASSIGNMENT COVERSHEET

Student ID : 219305860	**Student Name/ Names of all group members:** **AKASH DHITAL**
Programme: **BSC (Hons) Computer System Engineering**	**Module Code and Name:** CET341 Advance Database Technologies
Module Leader/ Module Tutor: **Ujiwal Mishra**	**Due Date:** 2022/07/08 **Hand in Date:** 2022/07/07
Assessment Title: **DBMS using Oracle and MongoDB**	
Learning Outcomes Assessed: (***number as appropriate*****)**	

	Mark
Areas for Commendation	
Areas for Improvement	
General Comments	

Assessor Signature :	**Overall mark (** subject to ratification by the assessment board**)**	**Moderator Signature**

……

I confirm that in submitting this assignment that I have read, understood and adhered to the University's Rules and procedures governing infringements of Assessment Regulations.

PRINT Student Name: _____AKASH DHITAL_____

Faculty Stamp ***(date/time)***

Student Signature : _____

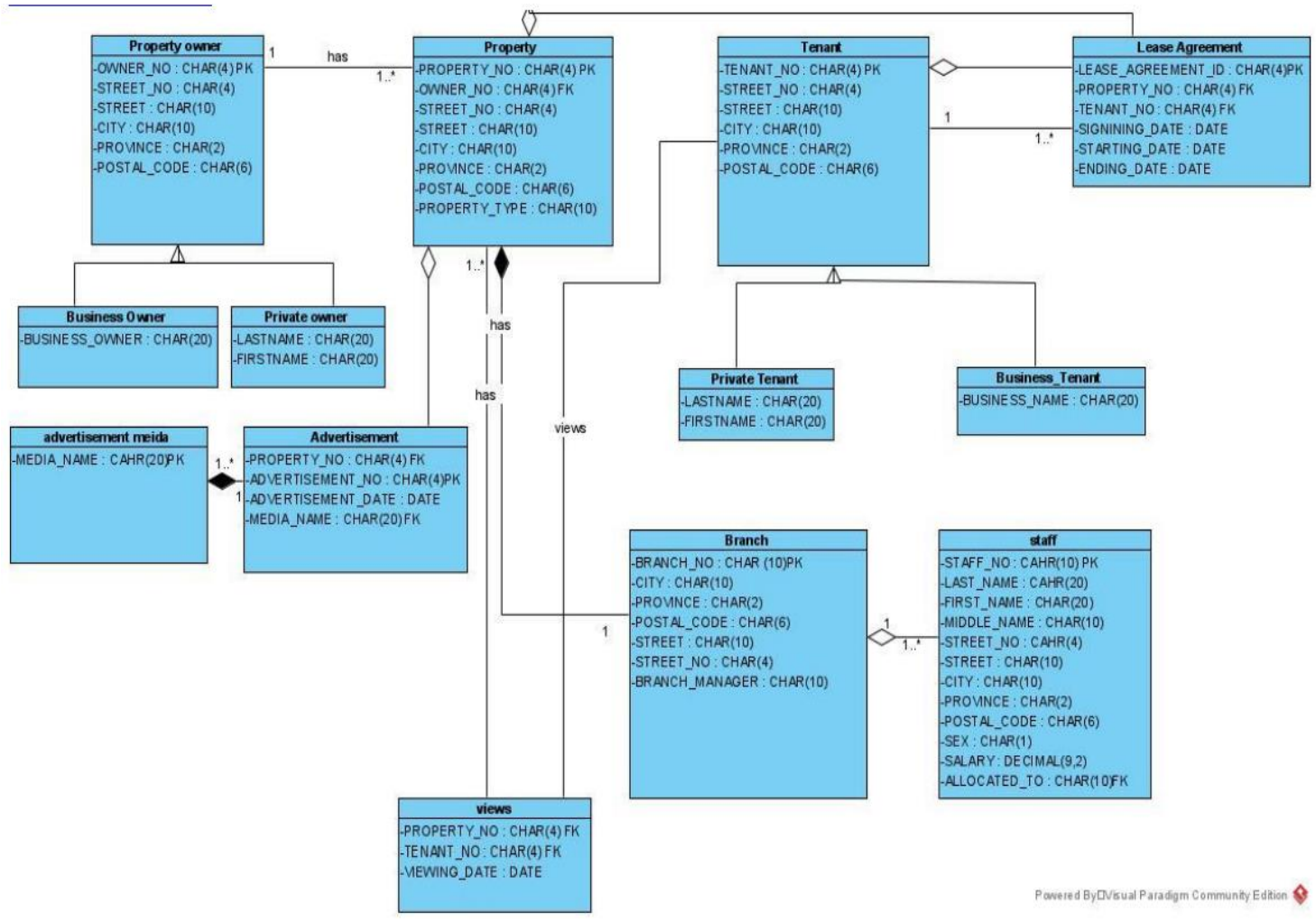
Module Code and Name: _____CET341 Advance Database Technologies_____

Name of Module Tutor : _____Ujiwal Mishra_____

CASE-STUDY:

Properties owned by both private and commercial entities are organized for rentals by R&K Industries. Each property owner has a unique owner number that serves as a means of identification, a record of the property's location (which includes a street, street number, town or city, and province), and the owner's name (comprising of first and last name for an individual or name of a business). Every property has a unique property number that serves as a record of its type, location, and history. Several advertisement might be placed for each property. A few dates might see each of these promotions appear in a variety of media. The advertisement mediums are identified by their unique names. Tenant is a reference to an individual or organization that approved a lease for a piece of property. Every lease has a distinctive, individual leasing agreement number. We keep a record of the lease agreement's signature date as well as its start and end dates. A tenant may rent out several properties.Before agreeing to a lease, a renter has the right to inspect the property more than once. A record of the name and address for each tenant. There is a specific number for each tenant. Each staff member at R&K Industries is assigned to a specific branch, and the company is organized into branches. One manager, who is also a staff member, oversees each branch. Each employee is identified by a specific staff number. A record of the name, number, sex, and salary for each employee is kept. One of the branches is in charge of each property. Each tenant speaks of the branch that looks after the space they are renting. One employee is responsible for overseeing each property. There is a location for each branch.

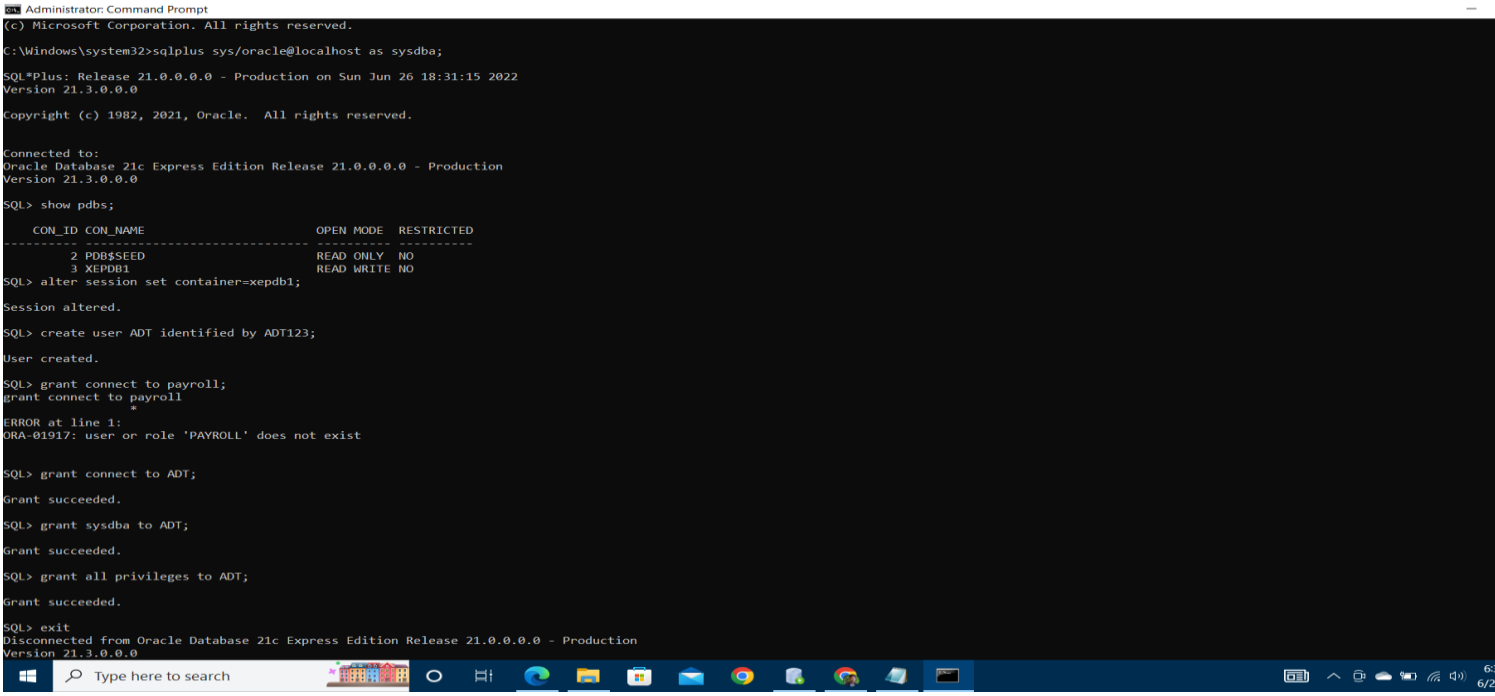
UML DIAGRAM:



THE CREATION OF THE DATABASE TABLES IN THE RELATED INSERT STATEMENTS

In development of a database it is a best practice to make sure that you execute the drop table statements to make sure that there are not any existing tables within the database you trying to create your tables in. The following screenshots will show the drop and creation of the tables within the database using Oracle databas.

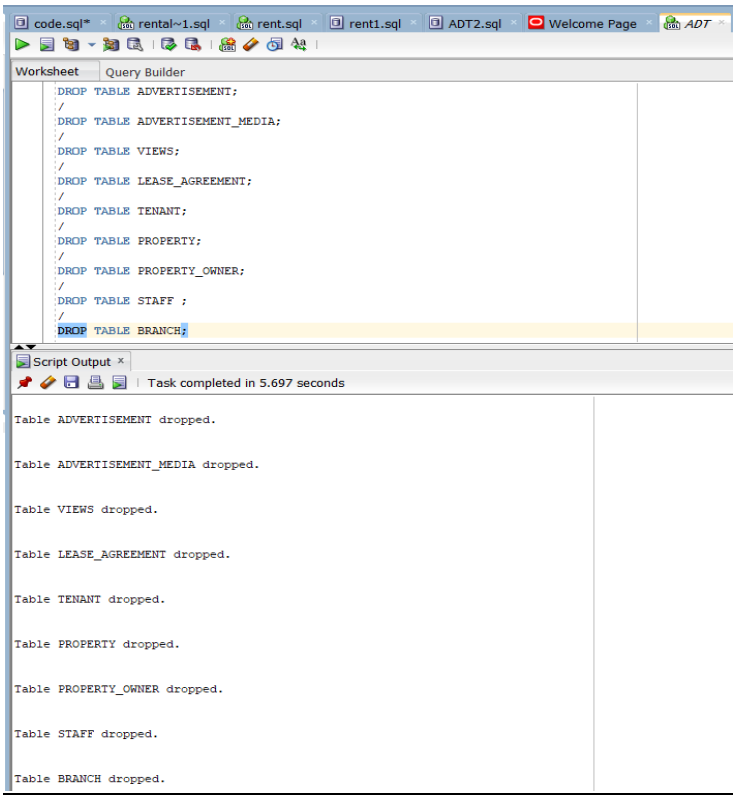
Database user creation:



SQL

Drop Table Statements

The below screen shows the dropping of tables and its returned results.



```
DROP TABLE PROPERTY_OWNER_ObjType;
DROP TYPE PROPERTY_OWNER_TYPE FORCE;
DROP TYPE LANDTYPE FORCE;
DROP TYPE NESTEDPROPERTY_OWNER_TYPE FORCE;
DROP TABLE LAND;
```

Script Output | Query Result

Task completed in 7.401 seconds

```
1 row inserted.

Table PROPERTY_OWNER_OBJTYPE dropped.

Type PROPERTY_OWNER_TYPE dropped.

Type LANDTYPE dropped.

Type NESTEDPROPERTY_OWNER_TYPE dropped.

Table LAND dropped.
```

Drop Types:

The below screen shows the dropping of types and its returned results.

code.sql* | rental~1.sql | rent.sql | rent1.sql | ADT2.sql | Welcome Page | ADT

Worksheet | Query Builder

```
DROP TYPE PROPERTY_OWNER_TYPE FORCE;
/
DROP TYPE PRIVATE_OWNER FORCE;
/
DROP TYPE BUSINESS_OWNER FORCE;
/
DROP TYPE TENANT_TYPE FORCE;
/
DROP TYPE PRIVATE_TENANT FORCE;
/
DROP TYPE BUSINESS_TENANT FORCE;
```

Script Output

Task completed in 2.092 seconds

```
Type PROPERTY_OWNER_TYPE dropped.

Type PRIVATE_OWNER dropped.

Type BUSINESS_OWNER dropped.

Type TENANT_TYPE dropped.

Type PRIVATE_TENANT dropped.

Type BUSINESS_TENANT dropped.
```

Type creation statements

An object type varies from local SQL datatypes in that it is client characterized, and it indicates both the fundamental persistent data and the connected practices.

```
CREATE TYPE PROPERTY_OWNER_TYPE AS OBJECT (
 OWNER_NO CHAR(4),
 STREET_NO CHAR(4),
 STREET CHAR(10),
 CITY CHAR(10),
 PROVINCE CHAR(20),
 POSTAL_CODE CHAR(6)
)NOT FINAL;

/
CREATE TYPE PRIVATE_OWNER UNDER PROPERTY_OWNER_TYPE(
FIRSTNAME CHAR(20),
LASTNAME CHAR(20)
);

CREATE TYPE BUSINESS_OWNER UNDER PROPERTY_OWNER_TYPE(

BUSINESS_OWNER_NAME CHAR(20)
);
```

Script Output ×

Task completed in 0.5 seconds

```
Type PROPERTY_OWNER_TYPE compiled


Type PRIVATE_OWNER compiled


Type PROPERTY_OWNER_TYPE dropped.


Type PRIVATE_OWNER dropped.


Type PROPERTY_OWNER_TYPE compiled


Type PRIVATE_OWNER compiled


Type BUSINESS_OWNER compiled
```

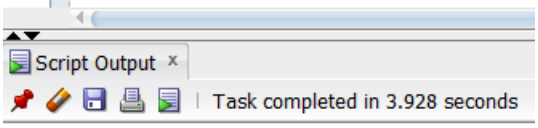
```
CREATE TYPE TENANT_TYPE AS OBJECT(
TENANT_NO CHAR(4),
STREET_NO CHAR(4),
STREET CHAR(10),
PROVINCE CHAR(2),
POSTAL_CODE CHAR(6)
)NOT FINAL;

/
CREATE TYPE PRIVATE_TENANT UNDER TENANT_TYPE(
FIRSTNAME CHAR(20),
LASTNAME CHAR(20)
);

/
CREATE TYPE BUSINESS_TENANT UNDER TENANT_TYPE (
BUSINESS_NAME CHAR(20)
);
```

Script Output ×

Task completed in 1.835 seconds

```
Type TENANT_TYPE compiled


Type PRIVATE_TENANT compiled


Type BUSINESS_TENANT compiled
```

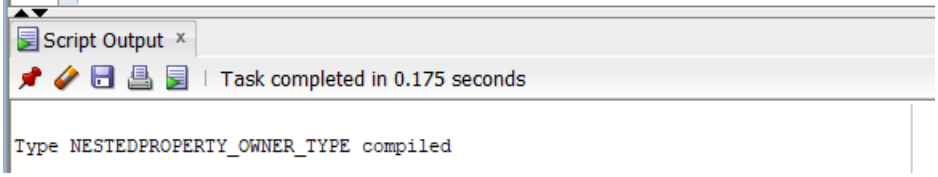
```
CREATE TYPE LANDTYPE AS OBJECT
( LAND_ID  NUMBER(5),
LAND_REG_NO  CHAR(9),
IS_AVAILABLE CHAR(3),
CATEGORY NESTEDPROPERTY_OWNER_TYPE
);
```




```
Type LANDTYPE compiled
```

Nested Property_owner_type:

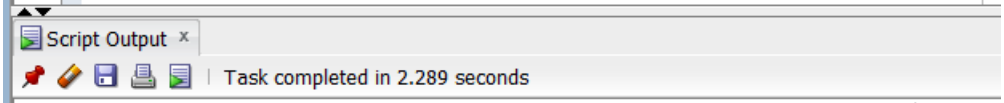
```
CREATE TYPE NESTEDPROPERTY_OWNER_TYPE AS TABLE OF REF PROPERTY_OWNER_TYPE;
```




```
Type NESTEDPROPERTY_OWNER_TYPE compiled
```

Creating Table Land of LandType:

```
CREATE TABLE LAND OF LANDTYPE
(
LAND_ID PRIMARY KEY,
LAND_REG_NO UNIQUE
)
NESTED TABLE category STORE AS
 NESTEDPROPERTY_OWNERTABLE;


 INSERT INTO LAND VALUES
(1, 'BA1KA4001', 'A', NESTEDPROPERTY_OWNER_TYPE());
INSERT INTO TABLE
(
 SELECT LD.category
 FROM LAND LD
 WHERE LD.LAND_ID = 1)
VALUES ((select ref(PT) from PROPERTY_OWNER_ObjType PT where OWNER_NO='0006'));
```




```
Table LAND created.


1 row inserted.


1 row inserted.
```

Creating and Insertion of data into PROPERTY_OWNER_objType(nested table):

```
CREATE TABLE PROPERTY_OWNER_ObjType of PROPERTY_OWNER_TYPE(
OWNER_NO NOT NULL,
PRIMARY KEY (OWNER_NO),
CHECK (PROVINCE IN ('BAGMATI','GANDAKI','SUDHURPASHCHIM','MADHESH','KARNALI','LUMBINI','PROVINCE_ONE')),
CHECK(REGEXP_LIKE(POSTAL_CODE,'[0-9][0-9][0-9]'))
);

insert into PROPERTY_OWNER_ObjType values (PROPERTY_OWNER_TYPE('O006','S140','HARINATH','BHAKTAPUR','BAGMATI','44351'));
insert into PROPERTY_OWNER_ObjType values (PRIVATE_OWNER('O014','S050','CHAKRAPATH','DOLAKHA','SUDHURPASHCHIM','44341','HARI','THAPA'));
insert into PROPERTY_OWNER_ObjType values (BUSINESS_OWNER('O015' ,'S140','HARINATH','BHAKTAPUR','BAGMATI','44351','RG INDUSTRIES'));
```

Script Output

Task completed in 5.037 seconds

```
Table PROPERTY_OWNER_OBJTYPE created.


1 row inserted.


1 row inserted.


1 row inserted.
```

Table creation and insertion statements

BRANCH TABLE:

```
CREATE TABLE BRANCH (
BRANCH_NO CHAR(10) NOT NULL,
STREET_NO CHAR(10) NOT NULL,
STREET CHAR(10) NOT NULL,
CITY CHAR(10) NOT NULL,
PROVINCE CHAR(20) NOT NULL,
POSTAL_CODE CHAR(6) NOT NULL,
BRANCH_MANAGER CHAR(10),
PRIMARY KEY (BRANCH_NO),
CHECK (PROVINCE IN ('BAGMATI','GANDAKI','SUDHURPASHCHIM','MADHESH','KARNALI','LUMBINI','PROVINCE_ONE')),
CHECK(REGEXP_LIKE(POSTAL_CODE,'[0-9][0-9][0-9]')),
UNIQUE(BRANCH_MANAGER)
);


INSERT INTO BRANCH(BRANCH_NO,STREET_NO,STREET,CITY,PROVINCE,POSTAL_CODE,BRANCH_MANAGER)
VALUES('B0001','00001','JYATHA','KATHMANDU','BAGMATI','44600','SANDESH');

INSERT INTO BRANCH(BRANCH_NO,STREET_NO,STREET,CITY,PROVINCE,POSTAL_CODE,BRANCH_MANAGER)
VALUES('B0002','00002','RAMSHAH','GORKHA','GANDAKI','34000','BIBEK');
```

Script Output

Task completed in 9.027 seconds

```
Table BRANCH created.


1 row inserted.


1 row inserted.
```

TABLE VIEW OF BRANCH:

	BRANCH_NO	STREET_NO	STREET	CITY	PROVINCE	POSTAL_CODE	BRANCH_MANAGER
1	B0001	00001	JYATHA	KATHMANDU	BAGMATI	44600	SANDESH
2	B0002	00002	RAMSHAH	GORKHA	GANDAKI	34000	BIBEK
3	B0003	00003	HARIPUR	SARLAHI	MADHESH	45800	ANCHALA
4	B0004	00004	BIRENDRA	JUMLA	KARNALI	21200	ALEX
5	B0005	00005	BASANTAPUR	KATHMANDU	BAGMATI	44601	GAURAV
6	B0006	00006	THAPATHALI	KATHMANDU	BAGMATI	44652	KIRAN

STAFF TABLE:

```
CREATE TABLE STAFF (
 STAFF_NO CHAR(10) NOT NULL,
 LAST_NAME CHAR(20) NOT NULL,
 FIRST_NAME CHAR(10) NOT NULL,
 MIDDLE_NAME CHAR(10),
 STREET_NO CHAR(4) NOT NULL,
 STREET CHAR(10) NOT NULL,
 CITY CHAR(10) NOT NULL,
 PROVINCE CHAR(20) NOT NULL,
 POSTAL_CODE CHAR(6) NOT NULL,
 SEX CHAR(1) NOT NULL,
 SALARY DECIMAL(9,2) NOT NULL,
 BRANCH_MANAGER CHAR(10) NOT NULL,
 BRANCH_NO CHAR(10) NOT NULL,
 PRIMARY KEY (STAFF_NO),
 FOREIGN KEY (BRANCH_NO) REFERENCES BRANCH,
  CHECK (PROVINCE IN ('BAGMATI','GANDAKI','SUDHURPASHCHIM','MADHESH','KARNALI','LUMBINI','PROVINCE_ONE')),
 CHECK (SEX IN ('F','M','N')),
 CHECK (SALARY > 0),
 CHECK(REGEXP_LIKE(POSTAL_CODE,'[0-9][0-9][0-9]'))
);

INSERT INTO STAFF(STAFF_NO, LAST_NAME, FIRST_NAME,MIDDLE_NAME,STREET_NO,STREET,CITY,PROVINCE,POSTAL_CODE,SEX,SALARY,BRANCH_MANAGER,BRANCH_NO)
VALUES('S001','SHARMA','SITA','DEVI','2011','JYATHA','KATHMANDU','BAGMATI','44600','F','5000.00','SANDESH','B0001');

INSERT INTO STAFF(STAFF_NO, LAST_NAME, FIRST_NAME,MIDDLE_NAME,STREET_NO,STREET,CITY,PROVINCE,POSTAL_CODE,SEX,SALARY,BRANCH_MANAGER,BRANCH_NO)
VALUES('S002','POUDEL','SUJAN','NA','2020','RAMSHAH','GORKHA','GANDAKI','34000','M','7000.00','BIBEK','B0002');
```

Script Output ×

Task completed in 4.737 seconds

```
Table STAFF created.


1 row inserted.


1 row inserted.


1 row inserted.
```

TABLE RESULT OF STAFF:

	STAFF_NO	LAST_NAME	FIRST_NAME	MIDDLE_NAME	STREET_NO	STREET	CITY	PROVINCE	POSTAL_CODE	SEX	SALARY	BRANCH_MANAGER	BRANCH_NO
1	S008	DHAKAL	ANISHA	KUMARI	0001	JYATHA	KATHMANDU	BAGMATI	44600	F	5000	SANDESH	B0001
2	S009	ARYAL	PRASHANT	RAJ	2020	BASANTAPUR	KATHMANDU	BAGMATI	44601	M	20000	BIBEK	B0002
3	S010	SHRESTHA	HARI	BAHADUR	1958	KALIKA	TANDI	BAGMATI	44800	M	50000	ANCHALA	B0003
4	S011	AMAGAI	SUNIL	PRASAD	1958	BIRENDRA	JUMLA	KARNALI	44800	M	60000	ALEX	B0004
5	S012	DANGOL	BALEN	BISHNU	1958	BIRENDRA	JUMLA	KARNALI	44800	M	70000	ALEX	B0004
6	S013	KC	ARJUN	NARSINGH	1958	BIRENDRA	JUMLA	KARNALI	44800	M	70000	ALEX	B0004
7	S001	SHARMA	SITA	DEVI	2011	JYATHA	KATHMANDU	BAGMATI	44600	F	5000	SANDESH	B0001
8	S002	POUDEL	SUJAN	NA	2020	RAMSHAH	GORKHA	GANDAKI	34000	M	7000	BIBEK	B0002
9	S005	THAPA	ABIRAL	BAHADUR	2020	BASANTAPUR	KATHMANDU	BAGMATI	44601	F	20000	AELSON	B0002
10	S003	DONG	LOK	BAHADUR	1958	KALIKA	CHITWAN	BAGMATI	44800	F	10000	ANCHALA	B0003
11	S004	MOTHOMELA	BOINEELO	LIFTUS	1955	BLOCK9	NAWALPUR	BAGMATI	44821	M	1000	ALEX	B0003
12	S007	SHRESTHA	MADAN	KRISHNA	1958	KALIKA	TANDI	BAGMATI	44800	M	50000	SUYOG	B0003

Creating Property_Owner Table :

```
CREATE TABLE PROPERTY_OWNER OF PROPERTY_OWNER_TYPE(
 OWNER_NO NOT NULL,
 PRIMARY KEY (OWNER_NO),
 CHECK (PROVINCE IN ('BAGMATI','GANDAKI','SUDHURPASHCHIM','MADHESH','KARNALI','LUMBINI','PROVINCE_ONE')),
 CHECK(REGEXP_LIKE(POSTAL_CODE,'[0-9][0-9][0-9]'))
 );

 INSERT INTO PROPERTY_OWNER
 VALUES(PRIVATE_OWNER('O001','S001','RAMSHAH','GORKHA','GANDAKI','34000','SANDESH','PANTA'));


 INSERT INTO PROPERTY_OWNER
 VALUES(PRIVATE_OWNER('O002','S010','BLOCK9','NAWALPUR','BAGMATI','44526','DONG','LOK'));


 INSERT INTO PROPERTY_OWNER
 VALUES(PRIVATE_OWNER('O003','S020','CHAKRAPATH','DOLAKHA','SUDHURPASHCHIM','44341','BIKASH','GIRI'));



 INSERT INTO PROPERTY_OWNER
 VALUES(BUSINESS_OWNER('O004','S120','BYPASS','BHARATPUR','BAGMATI','44371','DBLCARE'));

 INSERT INTO PROPERTY_OWNER
 VALUES(BUSINESS_OWNER('O005','S130','GATE','DHANKUTA','SUDHURPASHCHIM','44371','HAEL MINE'));

 INSERT INTO PROPERTY_OWNER
 VALUES(BUSINESS_OWNER('O006','S140','HARINATH','BHAKTAPUR','BAGMATI','44351','RG INDUSTRIES'));
```

Script Output | Task completed in 2.32 seconds

```
Table PROPERTY_OWNER created.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

Creating Property Table:

code.sql* | rental~1.sql | rent.sql | rent1.sql | Welcome Page | ADT

Worksheet | Query Builder

```
CREATE TABLE PROPERTY (
 PROPERTY_NO CHAR(4) NOT NULL,
 OWNER_NO CHAR(4) NOT NULL,
 STREET_NO CHAR(4) NOT NULL,
 STREET CHAR(10) NOT NULL,
 CITY CHAR(10) NOT NULL,
 PROVINCE CHAR(20) NOT NULL,
 POSTAL_CODE CHAR(6) NOT NULL,
 PROPERTY_TYPE CHAR (10),
 PRIMARY KEY (PROPERTY_NO),
 FOREIGN KEY (OWNER_NO) REFERENCES PROPERTY_OWNER,
CHECK (PROVINCE IN ('BAGMATI','GANDAKI','SUDHURPASHCHIM','MADHESH','KARNALI','LUMBINI','PROVINCE_ONE')),
 CHECK(REGEXP_LIKE(POSTAL_CODE,'[0-9][0-9][0-9]'))
);

INSERT INTO PROPERTY(PROPERTY_NO, OWNER_NO, STREET_NO,STREET , CITY, PROVINCE , POSTAL_CODE,PROPERTY_TYPE)
VALUES('P001','O001','2000','SNOW','HUMLA','KARNALI','44333','COMMERCIAL' );

INSERT INTO PROPERTY(PROPERTY_NO, OWNER_NO, STREET_NO,STREET , CITY, PROVINCE , POSTAL_CODE,PROPERTY_TYPE)
VALUES('P002','O001','2001','JANAKPUR','BARA','KARNALI','44833','RESIDENT' );


INSERT INTO PROPERTY(PROPERTY_NO, OWNER_NO, STREET_NO,STREET , CITY, PROVINCE , POSTAL_CODE,PROPERTY_TYPE)
VALUES('P003','O005','1830','KALI','NUWAKOT','BAGMATI','DFR558','COMMERCIAL');


INSERT INTO PROPERTY(PROPERTY_NO, OWNER_NO, STREET_NO,STREET , CITY, PROVINCE , POSTAL_CODE,PROPERTY_TYPE)
VALUES('P004','O003','1825','SNOW','HUMLA','KARNALI','44330','RESIDENT');
```

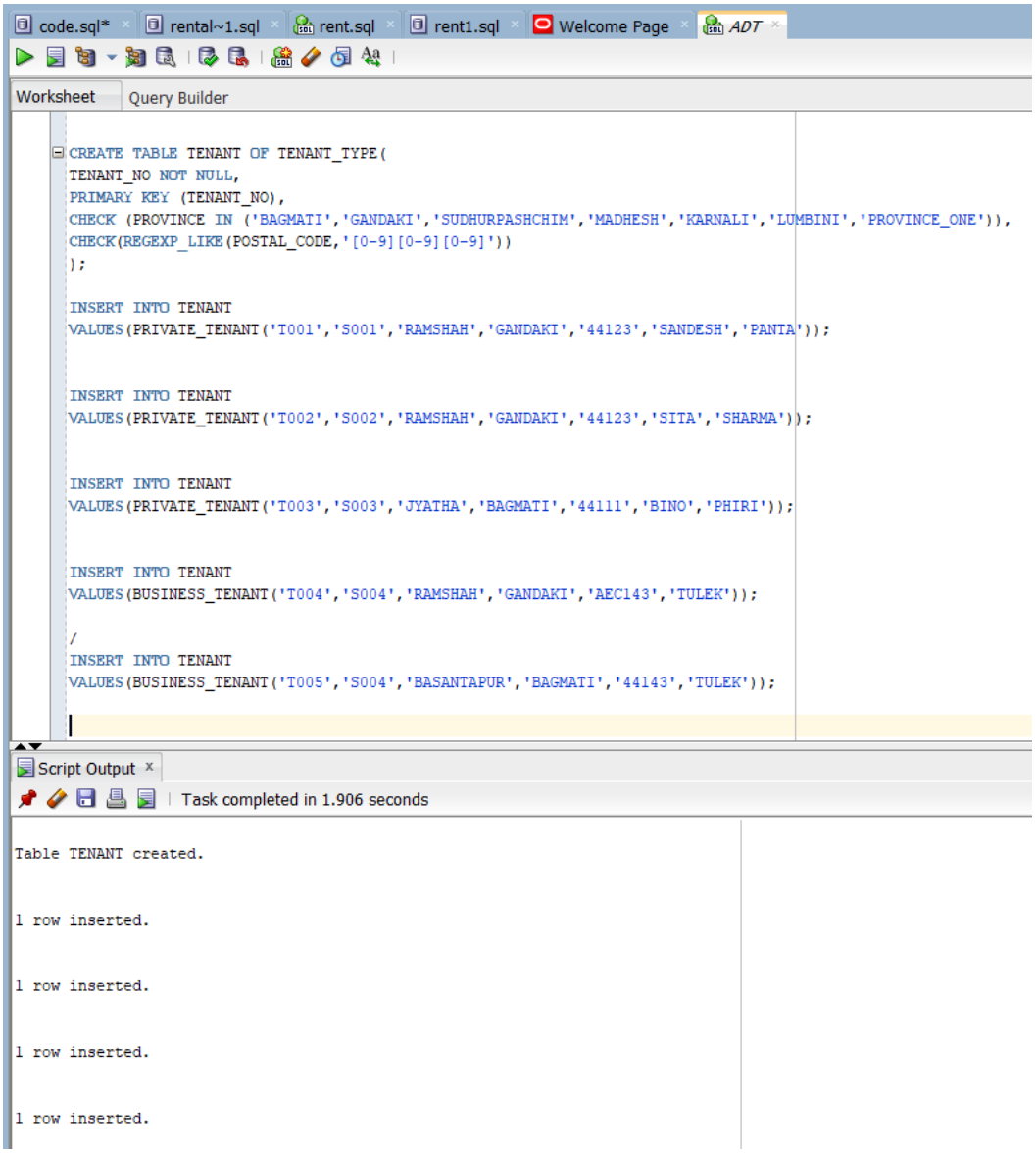
Script Output | Task completed in 6.084 seconds

```
Table PROPERTY created.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

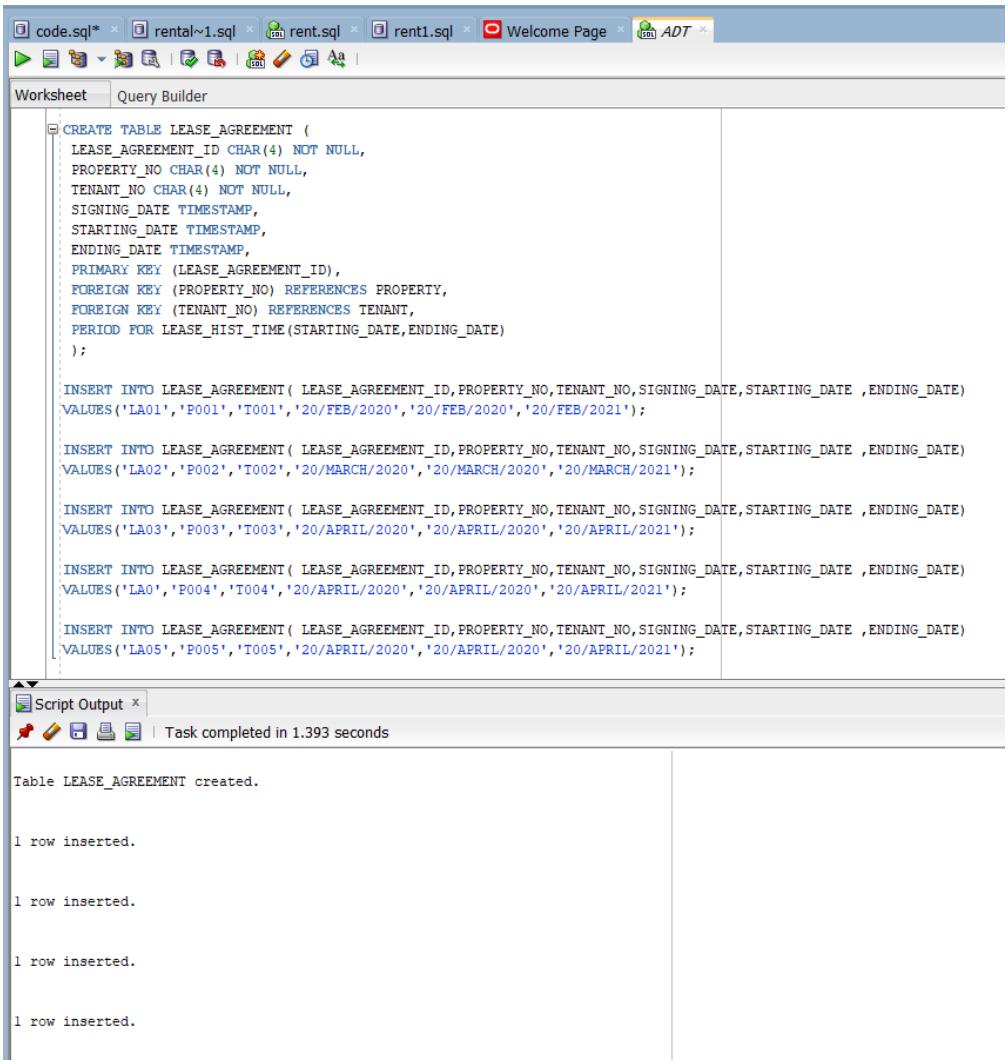
Result of Property:

	PROPERTY_NO	OWNER_NO	STREET_NO	STREET	CITY	PROVINCE	POSTAL_CODE	PROPERTY_TYPE
1	P001	O001	2000	SNOW	HUMLA	KARNALI	44333	COMMERCIAL
2	P002	O001	2001	JANAKPUR	BARA	KARNALI	44833	RESIDENT
3	P003	O005	1830	KALI	NUWAKOT	BAGMATI	DFR558	COMMERCIAL
4	P004	O003	1825	SNOW	HUMLA	KARNALI	44330	RESIDENT
5	P005	O001	2011	BLOCK9	NAWALPUR	BAGMATI	44123	COMMERCIAL
6	P006	O005	1830	RIO	GULMI	KARNALI	44558	COMMERCIAL
7	P008	O011	2000	SNOW	HUMLA	KARNALI	44333	COMMERCIAL
8	P007	O005	1830	RIO	GULMI	KARNALI	44558	COMMERCIAL

Creating Table of Tenant:



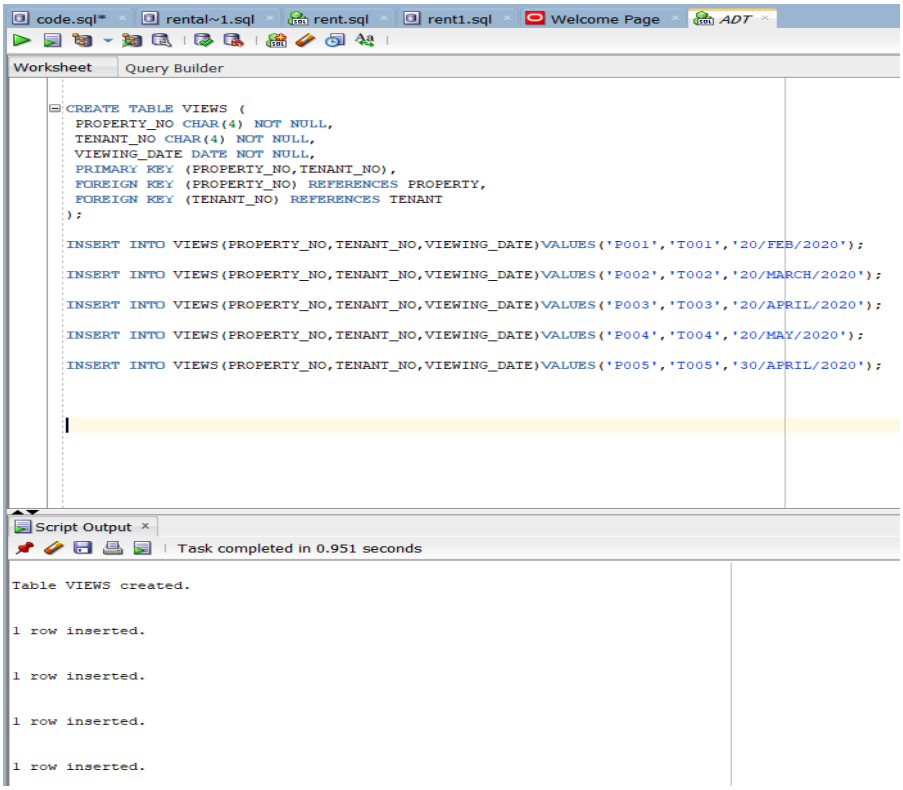
LEASE_AGREEMENT TABLE:



```
CREATE TABLE LEASE_AGREEMENT (
 LEASE_AGREEMENT_ID CHAR(4) NOT NULL,
 PROPERTY_NO CHAR(4) NOT NULL,
 TENANT_NO CHAR(4) NOT NULL,
 SIGNING_DATE TIMESTAMP,
 STARTING_DATE TIMESTAMP,
 ENDING_DATE TIMESTAMP,
 PRIMARY KEY (LEASE_AGREEMENT_ID),
 FOREIGN KEY (PROPERTY_NO) REFERENCES PROPERTY,
 FOREIGN KEY (TENANT_NO) REFERENCES TENANT,
 PERIOD FOR LEASE_HIST_TIME(STARTING_DATE,ENDING_DATE)
 );

INSERT INTO LEASE_AGREEMENT( LEASE_AGREEMENT_ID,PROPERTY_NO,TENANT_NO,SIGNING_DATE,STARTING_DATE ,ENDING_DATE)
VALUES('LA01','P001','T001','20/FEB/2020','20/FEB/2020','20/FEB/2021');

INSERT INTO LEASE_AGREEMENT( LEASE_AGREEMENT_ID,PROPERTY_NO,TENANT_NO,SIGNING_DATE,STARTING_DATE ,ENDING_DATE)
VALUES('LA02','P002','T002','20/MARCH/2020','20/MARCH/2020','20/MARCH/2021');

INSERT INTO LEASE_AGREEMENT( LEASE_AGREEMENT_ID,PROPERTY_NO,TENANT_NO,SIGNING_DATE,STARTING_DATE ,ENDING_DATE)
VALUES('LA03','P003','T003','20/APRIL/2020','20/APRIL/2020','20/APRIL/2021');

INSERT INTO LEASE_AGREEMENT( LEASE_AGREEMENT_ID,PROPERTY_NO,TENANT_NO,SIGNING_DATE,STARTING_DATE ,ENDING_DATE)
VALUES('LA0','P004','T004','20/APRIL/2020','20/APRIL/2020','20/APRIL/2021');

INSERT INTO LEASE_AGREEMENT( LEASE_AGREEMENT_ID,PROPERTY_NO,TENANT_NO,SIGNING_DATE,STARTING_DATE ,ENDING_DATE)
VALUES('LA05','P005','T005','20/APRIL/2020','20/APRIL/2020','20/APRIL/2021');
```

```
Table LEASE_AGREEMENT created.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

Views Table:



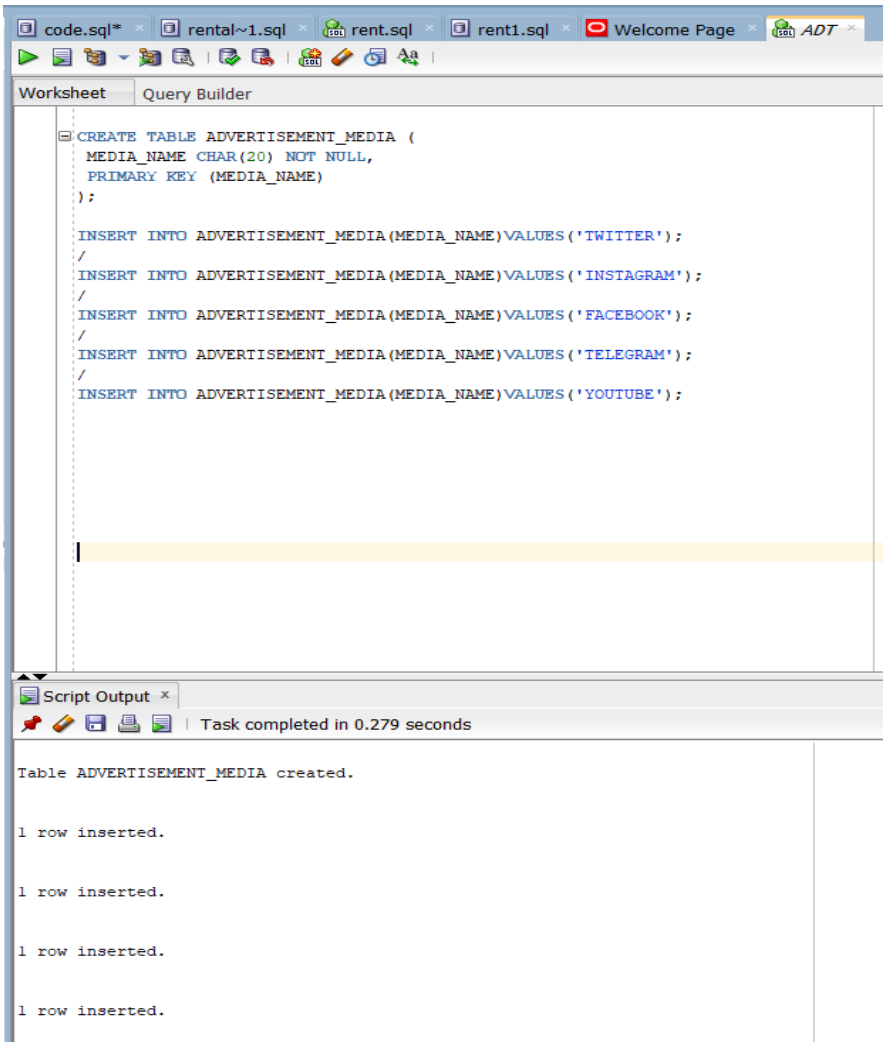
```
CREATE TABLE VIEWS (
 PROPERTY_NO CHAR(4) NOT NULL,
 TENANT_NO CHAR(4) NOT NULL,
 VIEWING_DATE DATE NOT NULL,
 PRIMARY KEY (PROPERTY_NO,TENANT_NO),
 FOREIGN KEY (PROPERTY_NO) REFERENCES PROPERTY,
 FOREIGN KEY (TENANT_NO) REFERENCES TENANT
);

INSERT INTO VIEWS(PROPERTY_NO,TENANT_NO,VIEWING_DATE)VALUES('P001','T001','20/FEB/2020');

INSERT INTO VIEWS(PROPERTY_NO,TENANT_NO,VIEWING_DATE)VALUES('P002','T002','20/MARCH/2020');

INSERT INTO VIEWS(PROPERTY_NO,TENANT_NO,VIEWING_DATE)VALUES('P003','T003','20/APRIL/2020');

INSERT INTO VIEWS(PROPERTY_NO,TENANT_NO,VIEWING_DATE)VALUES('P004','T004','20/MAY/2020');

INSERT INTO VIEWS(PROPERTY_NO,TENANT_NO,VIEWING_DATE)VALUES('P005','T005','30/APRIL/2020');
```

```
Table VIEWS created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

Result of VIEWS:

	PROPERTY_NO	TENANT_NO	VIEWING_DATE
1	P001	T001	20-FEB-20
2	P002	T002	20-MAR-20
3	P003	T003	20-APR-20
4	P004	T004	20-MAY-20
5	P005	T005	30-APR-20

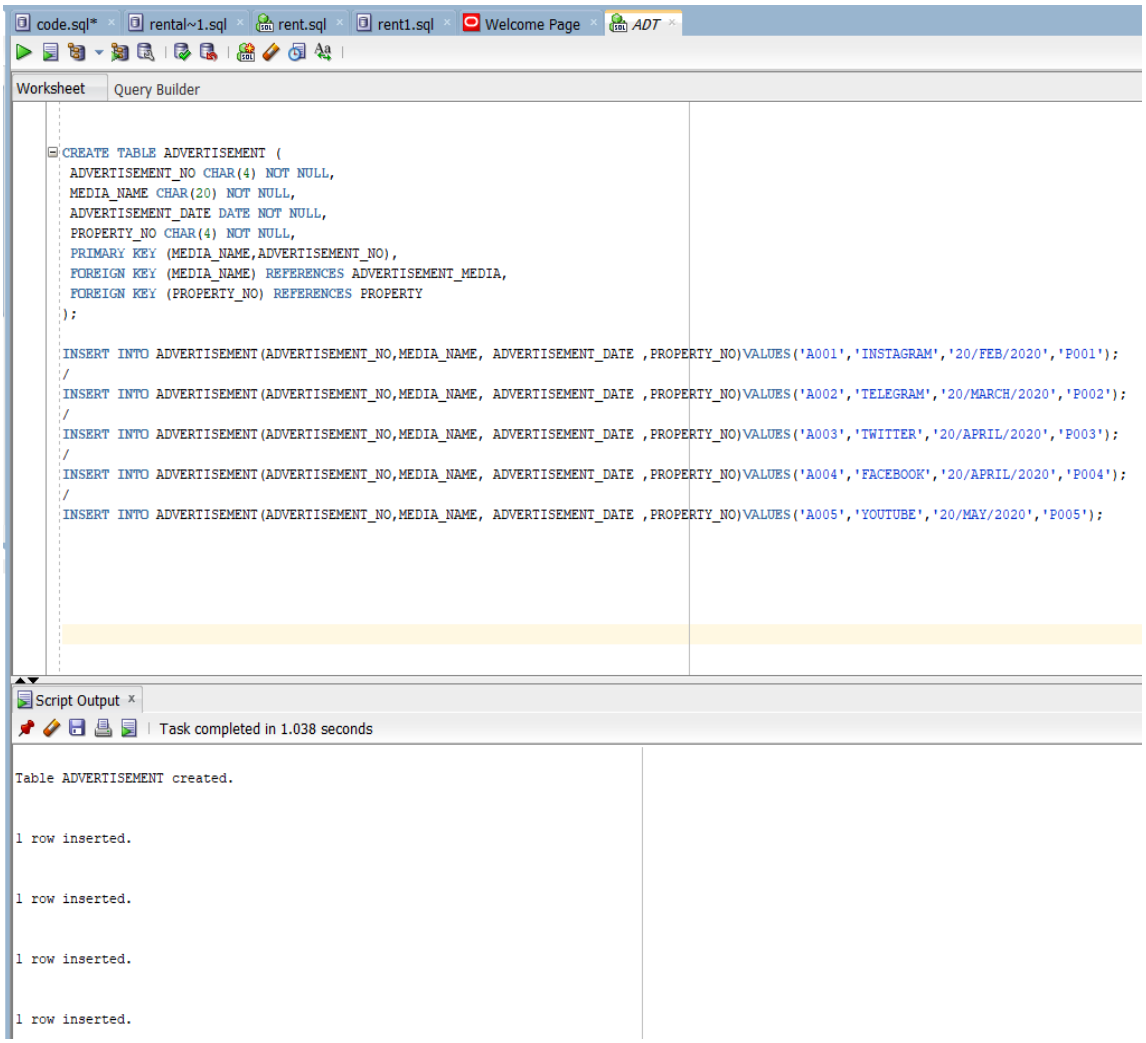
Advertisement Media Table:



Result of Advertisement Media:

	MEDIA_NAME
1	FACEBOOK
2	INSTAGRAM
3	TELEGRAM
4	TWITTER
5	YOUTUBE

Advertisement Table:



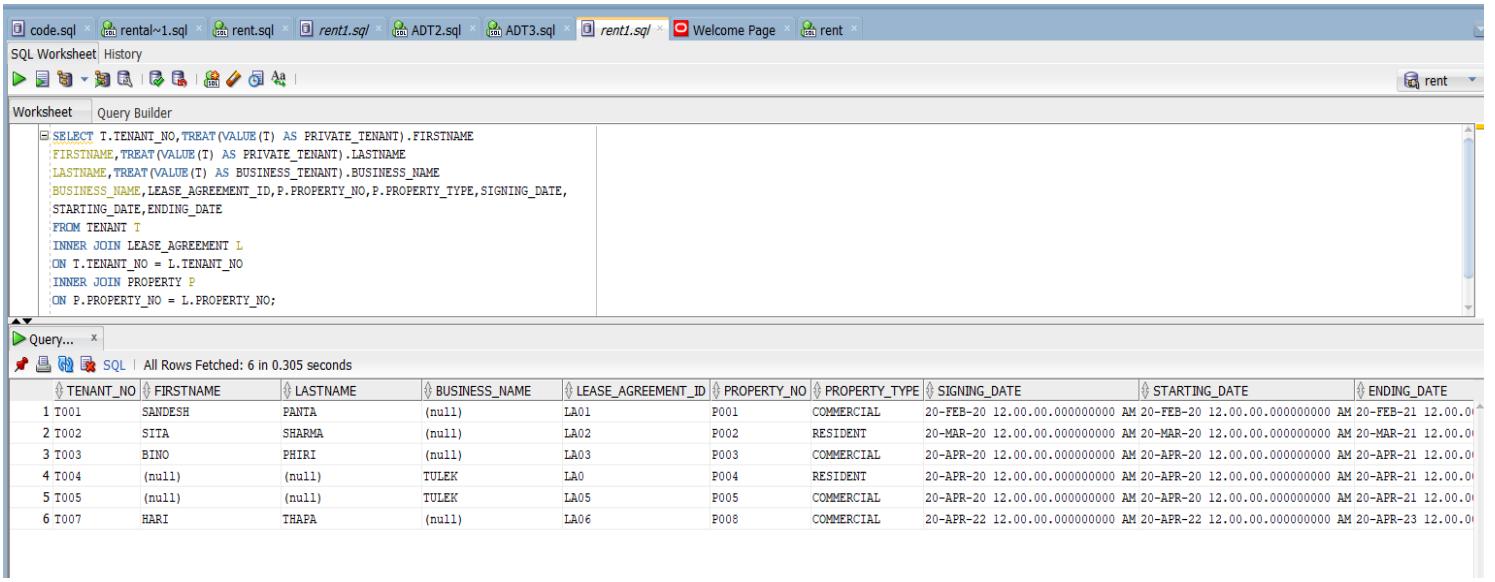
Result of Advertisement:

	ADVERTISEMENT_NO	MEDIA_NAME	ADVERTISEMENT_DATE	PROPERTY_NO
1	A001	INSTAGRAM	20-FEB-20	P001
2	A002	TELEGRAM	20-MAR-20	P002
3	A003	TWITTER	20-APR-20	P003
4	A004	FACEBOOK	20-APR-20	P004
5	A005	YOUTUBE	20-MAY-20	P005

SQL QUERIES AND OUTPUT:

--QUERY A: JOIN THREE OR MORE TABLES—

The INNER JOIN chooses all rows from 3 tables if there is a match between the column property number in all the tables it will join based on that.



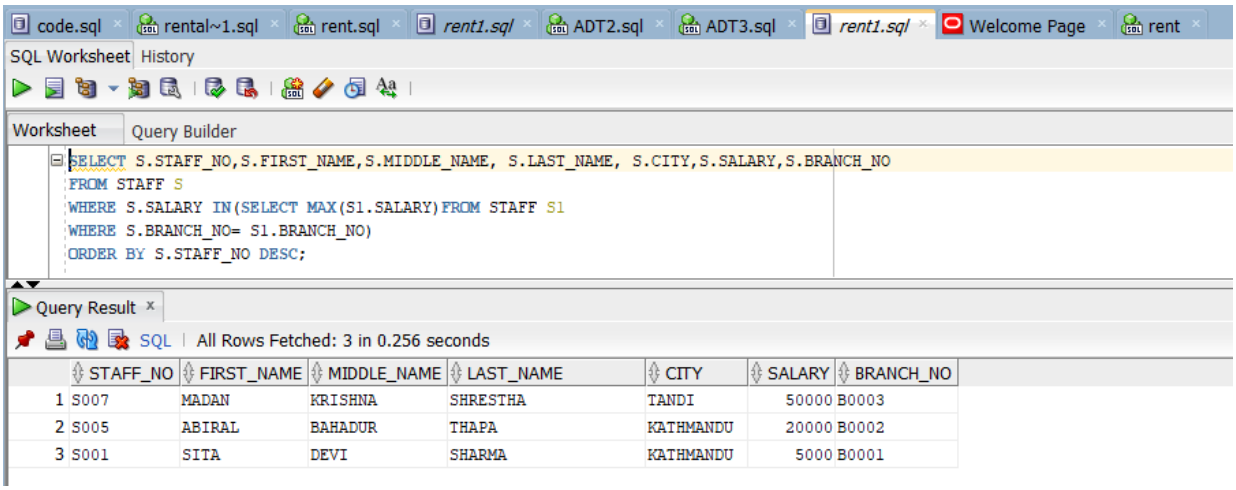
```
SELECT T.TENANT_NO,TREAT(VALUE(T) AS PRIVATE_TENANT).FIRSTNAME
FIRSTNAME,TREAT(VALUE(T) AS PRIVATE_TENANT).LASTNAME
LASTNAME,TREAT(VALUE(T) AS BUSINESS_TENANT).BUSINESS_NAME
BUSINESS_NAME,LEASE_AGREEMENT_ID,P.PROPERTY_NO,P.PROPERTY_TYPE,SIGNING_DATE,
STARTING_DATE,ENDING_DATE
FROM TENANT T
INNER JOIN LEASE_AGREEMENT L
ON T.TENANT_NO = L.TENANT_NO
INNER JOIN PROPERTY P
ON P.PROPERTY_NO = L.PROPERTY_NO;
```

All Rows Fetched: 6 in 0.305 seconds

	TENANT_NO	FIRSTNAME	LASTNAME	BUSINESS_NAME	LEASE_AGREEMENT_ID	PROPERTY_NO	PROPERTY_TYPE	SIGNING_DATE	STARTING_DATE	ENDING_DATE
1	T001	SANDESH	PANTA	(null)	LA01	P001	COMMERCIAL	20-FEB-20 12.00.00.000000000 AM	20-FEB-20 12.00.00.000000000 AM	20-FEB-21 12.00.0
2	T002	SITA	SHARMA	(null)	LA02	P002	RESIDENT	20-MAR-20 12.00.00.000000000 AM	20-MAR-20 12.00.00.000000000 AM	20-MAR-21 12.00.0
3	T003	BINO	PHIRI	(null)	LA03	P003	COMMERCIAL	20-APR-20 12.00.00.000000000 AM	20-APR-20 12.00.00.000000000 AM	20-APR-21 12.00.0
4	T004	(null)	(null)	TULEK	LA0	P004	RESIDENT	20-APR-20 12.00.00.000000000 AM	20-APR-20 12.00.00.000000000 AM	20-APR-21 12.00.0
5	T005	(null)	(null)	TULEK	LA05	P005	COMMERCIAL	20-APR-20 12.00.00.000000000 AM	20-APR-20 12.00.00.000000000 AM	20-APR-21 12.00.0
6	T007	HARI	THAPA	(null)	LA06	P008	COMMERCIAL	20-APR-22 12.00.00.000000000 AM	20-APR-22 12.00.00.000000000 AM	20-APR-23 12.00.0

MAKE USE OF SUB-QUERIES

The following sub-query will display staff with the highest salary in each branch

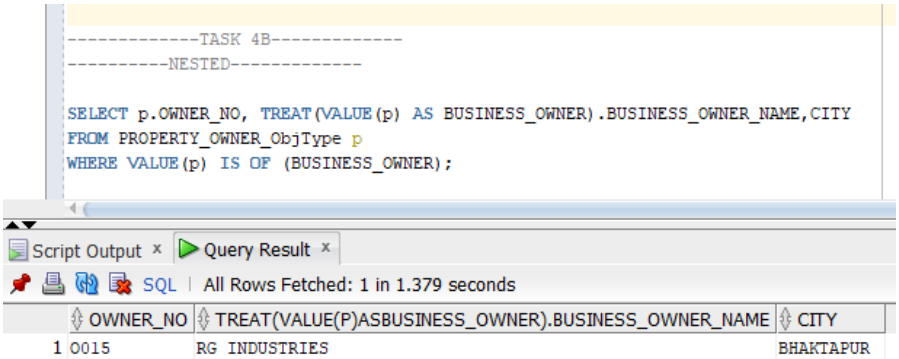


```
SELECT S.STAFF_NO,S.FIRST_NAME,S.MIDDLE_NAME, S.LAST_NAME, S.CITY,S.SALARY,S.BRANCH_NO
FROM STAFF S
WHERE S.SALARY IN(SELECT MAX(S1.SALARY)FROM STAFF S1
WHERE S.BRANCH_NO= S1.BRANCH_NO)
ORDER BY S.STAFF_NO DESC;
```

All Rows Fetched: 3 in 0.256 seconds

	STAFF_NO	FIRST_NAME	MIDDLE_NAME	LAST_NAME	CITY	SALARY	BRANCH_NO
1	S007	MADAN	KRISHNA	SHRESTHA	TANDI	50000	B0003
2	S005	ABIRAL	BAHADUR	THAPA	KATHMANDU	20000	B0002
3	S001	SITA	DEVI	SHARMA	KATHMANDU	5000	B0001

MAKE USE OF NESTED TABLE Query:



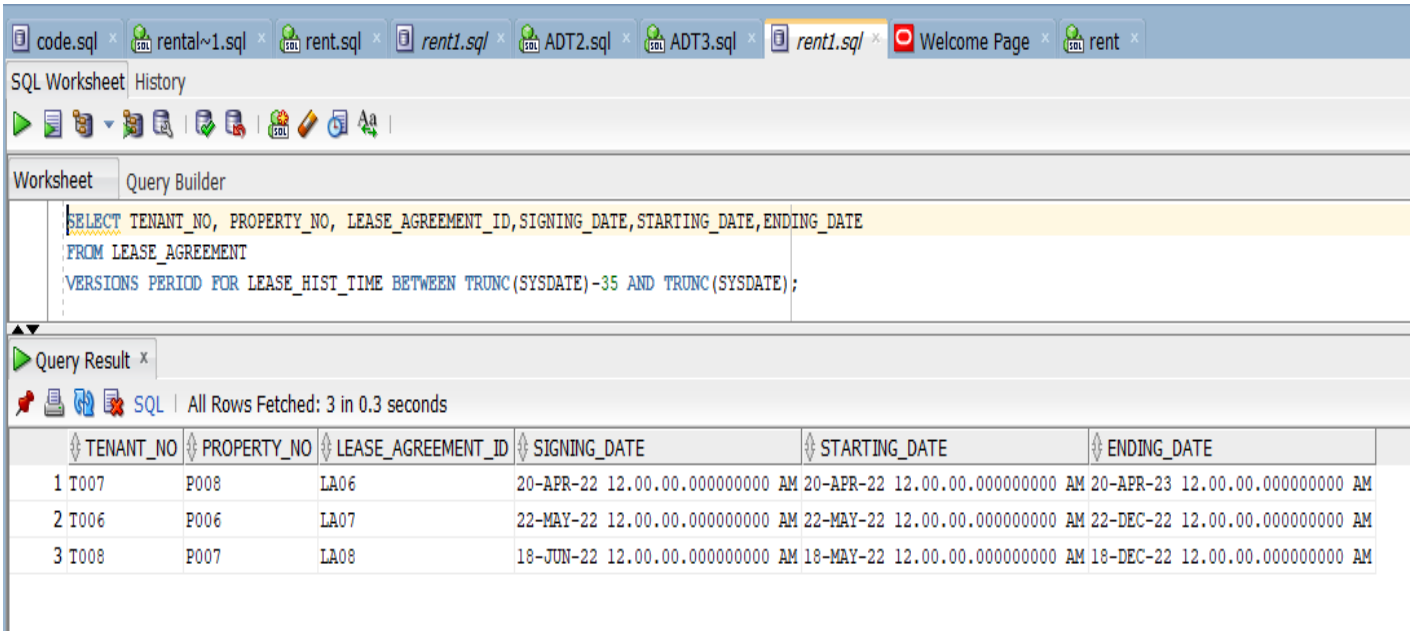
```
-------------TASK 4B-------------
----------NESTED-------------

SELECT p.OWNER_NO, TREAT(VALUE(p) AS BUSINESS_OWNER).BUSINESS_OWNER_NAME,CITY
FROM PROPERTY_OWNER_ObjType p
WHERE VALUE(p) IS OF (BUSINESS_OWNER);
```

All Rows Fetched: 1 in 1.379 seconds

	OWNER_NO	TREAT(VALUE(P)ASBUSINESS_OWNER).BUSINESS_OWNER_NAME	CITY
1	0015	RG INDUSTRIES	BHAKTAPUR

MAKE USE OF TEMPORAL FEATURES

This query will display an agreement that was made and entered in the last 35-day equivalent to 5weeks.

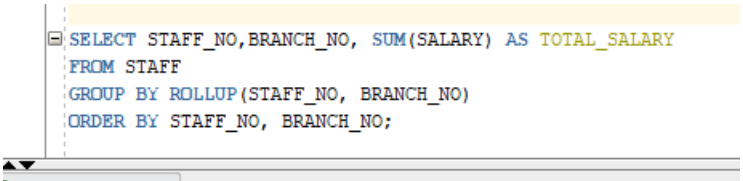


	TENANT_NO	PROPERTY_NO	LEASE_AGREEMENT_ID	SIGNING_DATE	STARTING_DATE	ENDING_DATE
1	T007	P008	LA06	20-APR-22 12.00.00.000000000 AM	20-APR-22 12.00.00.000000000 AM	20-APR-23 12.00.00.000000000 AM
2	T006	P006	LA07	22-MAY-22 12.00.00.000000000 AM	22-MAY-22 12.00.00.000000000 AM	22-DEC-22 12.00.00.000000000 AM
3	T008	P007	LA08	18-JUN-22 12.00.00.000000000 AM	18-MAY-22 12.00.00.000000000 AM	18-DEC-22 12.00.00.000000000 AM

MAKE USE OF OLAP FEATURES

-- --ROLLUP—

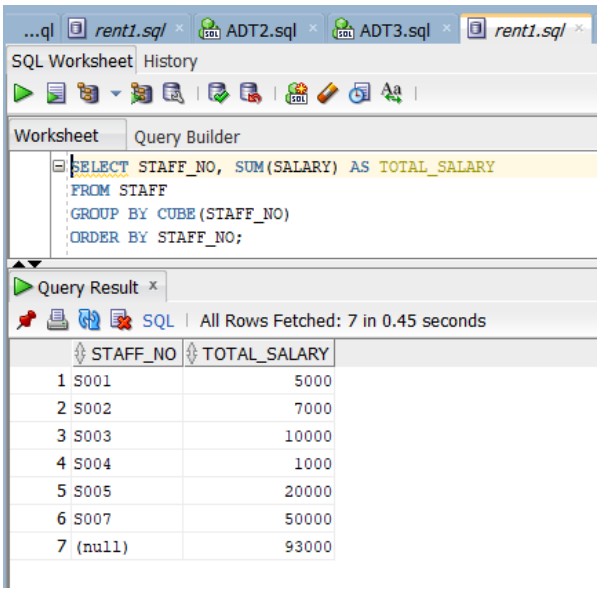
The roll up query aggregates the grand totals and groups them using the staff number and branch number.



	STAFF_NO	BRANCH_NO	TOTAL_SALARY
1	S001	B0001	5000
2	S001	(null)	5000
3	S002	B0002	7000
4	S002	(null)	7000
5	S003	B0003	10000
6	S003	(null)	10000
7	S004	B0003	1000
8	S004	(null)	1000
9	S005	B0002	20000
10	S005	(null)	20000
11	S007	B0003	50000
12	S007	(null)	50000
13	(null)	(null)	93000

----CUBE---

The cube group by differs from the roll up since the cub will generate the subtotals of all combinations of the specified columns.



	STAFF_NO	TOTAL_SALARY
1	S001	5000
2	S002	7000
3	S003	10000
4	S004	1000
5	S005	20000
6	S007	50000
7	(null)	93000

MONGODB

Creating MongoDB database that will be used to store the related collections and its documents. The following shows how all that was implemented with the use screenshots.

Database Creation and Creating Collections that correspond to the SQL tables in MONGODB

```
---
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
rent    0.000GB
rental  0.000GB
> use ADT
switched to db ADT
> db.createCollection("BRANCH")
{ "ok" : 1 }
> db.createCollection("PROPERTY")
{ "ok" : 1 }
> db.createCollection("ADVERTISEMENT")
{ "ok" : 1 }
>
```

Inserting documents into the collections

Branch collection:

```
> db.BRANCH.insert({
... "BRANCH_NO" : "B001",
...     "ADDRESS" : {
...         "STREET_NO" : "00001",
...         "STREET" : "JYATHA",
...         "CITY" : "KATHMANDU",
...         "PROVINCE" : "BAGMATI",
...         "POSTAL_CODE" : "44600"
...     },
...     "BRANCH_MANAGER" : "SANDESH",
...     "STAFF" : [
...         {
...             "STAFF_NO" : "S001",
...             "LAST_NAME" : "SHARMA",
...             "FIRST_NAME" : "SITA",
...             "MIDDLE_NAME" : "DEVI",
...             "SEX" : "F",
...             "SALARY" : 5000.0
...         },
...         {
...             "STAFF_NO" : "S008",
...             "LAST_NAME" : "DHAKAL",
...             "FIRST_NAME" : "ANISHA",
...             "MIDDLE_NAME" : "KUMARI",
...             "SEX" : "F",
...             "SALARY" : 5000.0
...         }
...     ]
...
... })
WriteResult({ "nInserted" : 1 })
```

```
> db.BRANCH.insert({
...   "BRANCH_NO" : "B002",
...      "ADDRESS" : {
...          "STREET_NO" : "00002",
...          "STREET" : "RAMSHAH",
...          "CITY" : "GORKHA",
...          "PROVINCE" : "GANDAKI",
...          "POSTAL_CODE" : "ABC111"
...      },
...      "BRANCH_MANAGER" : "BIBEK",
...      "STAFF" : [
...          {
...              "STAFF_NO" : "S002",
...              "LAST_NAME" : "POUDEL",
...              "FIRST_NAME" : "SUJAN",
...              "MIDDLE_NAME" : "NA",
...              "SEX" : "M",
...              "SALARY" : 7000.0
...          },
...          {
...              "STAFF_NO" : "S005",
...              "LAST_NAME" : "THAPA",
...              "FIRST_NAME" : "ABIRAL",
...              "MIDDLE_NAME" : "BAHADUR",
...              "SEX" : "M",
...              "SALARY" : 200000.0
...          },
...          {
...              "STAFF_NO" : "S009",
...              "LAST_NAME" : "ARYAL",
...              "FIRST_NAME" : "PRASHANT",
...              "MIDDLE_NAME" : "RAJ",
...              "SEX" : "M",
...              "SALARY" : 200000.0
...          }
...      ]
... })
WriteResult({ "nInserted" : 1 })
>
```

```
> db.BRANCH.insert({
...   "BRANCH_NO" : "B003",
...      "ADDRESS" : {
...          "STREET_NO" : "00003",
...          "STREET" : "HARIPUR",
...          "CITY" : "SARLAHI",
...          "PROVINCE" : "MADHESH",
...          "POSTAL_CODE" : "45800"
...      },
...      "BRANCH_MANAGER" : "ANCHALA",
...      "STAFF" : [
...          {
...              "STAFF_NO" : "S003",
...              "LAST_NAME" : "DONG",
...              "FIRST_NAME" : "LOK",
...              "MIDDLE_NAME" : "BAHADUR",
...              "SEX" : "M",
...              "SALARY" : 100000.0
...          },
...          {
...              "STAFF_NO" : "S007",
...              "LAST_NAME" : "SHRESTHA",
...              "FIRST_NAME" : "MADAN",
...              "MIDDLE_NAME" : "KRISHNA",
...              "SEX" : "M",
...              "SALARY" : 200000.0
...          },
...          {
...              "STAFF_NO" : "S010",
...              "LAST_NAME" : "SHRESTHA",
...              "FIRST_NAME" : "HARI",
...              "MIDDLE_NAME" : "BAHADUR",
...              "SEX" : "M",
...              "SALARY" : 200000.0
...          }
...      ]
...
... })
WriteResult({ "nInserted" : 1 })
>
```

Property collection:

```
>  db.PROPERTY.insert({
...
...     "PROPERTY_NO" : "P001",
...      "PROPERTY_TYPE" : "RESIDENTAL",
...      "PROPERTY_OWNER" : {
...          "OWNER_NO" : "O001",
...          "STREET_NO" : "S001",
...          "STREET" : "JYATHA",
...          "CITY" : "KATHMANDU",
...          "PROVINCE" : "BAGMATI",
...          "POSTAL_CODE" : "44600",
...          "PRIVATE_OWNER" : {
...              "FIRSTNAME" : "SANDESH",
...              "LASTNAME" : "PANTA"
...          }
...      },
...      "LEASE_AGREEMENT" : {
...          "LEASE_AGREEMENT_ID" : "LA01",
...          "SIGNINING_DATE" : ISODate("2021-10-02T01:11:18.965+0000"),
...          "STARTING_DATE" : ISODate("2021-10-02T01:11:18.965+0000"),
...          "ENDING_DATE" : ISODate("2021-10-02T01:11:18.965+0000")
...      },
...      "TENANT" : {
...          "TENANT_NO" : "T001",
...          "PRIVATE_TENANT" : {
...              "FIRSTNAME" : "SANDESH",
...              "LASTNAME" : "PANTA"
...          }
...      },
...      "BRANCH_NO" : "B001"
... })
WriteResult({ "nInserted" : 1 })
>
```

```
> db.PROPERTY.insert({
...      "PROPERTY_NO" : "P002",
...      "PROPERTY_TYPE" : "COMMERCIAL",
...      "PROPERTY_OWNER" : {
...          "OWNER_NO" : "O002",
...          "STREET_NO" : "2020",
...          "STREET" : "RAMSHAH",
...          "CITY" : "GORKHA",
...          "PROVINCE" : "GANDAKI",
...          "POSTAL_CODE" : "ABC111",
...          "BUISINESS_OWNER" : {
...              "BUSINESS_OWNER_NAME" : "TULEK"
...          }
...      },
...      "LEASE_AGREEMENT" : {
...          "LEASE_AGREEMENT_ID" : "LA02",
...          "SIGNINING_DATE" : ISODate("2021-12-02T01:11:18.965+0000"),
...          "STARTING_DATE" : ISODate("2021-12-02T01:11:18.965+0000"),
...          "ENDING_DATE" : ISODate("2022-12-02T01:11:18.965+0000")
...      },
...      "TENANT" : {
...          "TENANT_NO" : "T002",
...          "BUSINESS_TENANT" : {
...              "BUSINESS_NAME" : "DBL CARE"
...          }
...      },
...      "BRANCH_NO" : "B002"
...
... })
WriteResult({ "nInserted" : 1 })
>
```

```
> db.PROPERTY.insert({
...
... "PROPERTY_NO" : "P003",
...     "PROPERTY_TYPE" : "INDUSTRIAL",
...     "PROPERTY_OWNER" : {
...         "OWNER_NO" : "O003",
...         "STREET_NO" : "S005",
...         "STREET" : "JYATHA",
...         "CITY" : "KATHMANDU",
...         "PROVINCE" : "BAGMATI",
...         "POSTAL_CODE" : "44600",
...         "BUISINESS_OWNER" : {
...             "BUSINESS_OWNER_NAME" : "AKASH"
...         }
...     },
...     "LEASE_AGREEMENT" : {
...         "LEASE_AGREEMENT_ID" : "LA03",
...         "SIGNINING_DATE" : ISODate("2021-10-02T01:11:18.965+0000"),
...         "STARTING_DATE" : ISODate("2021-10-02T01:11:18.965+0000"),
...         "ENDING_DATE" : ISODate("2022-10-02T01:11:18.965+0000")
...     },
...     "TENANT" : {
...         "TENANT_NO" : "T003",
...         "BUSINESS_TENANT" : {
...             "BUSINESS_NAME" : "RG INDUSTRIES"
...         }
...     },
...     "BRANCH_NO" : "B003"
... })
WriteResult({ "nInserted" : 1 })
>
```

Advertisement collection:

```
> db.ADVERTISEMENT.insert({
...     "ADVERTISEMENT_NO" : "A001",
...     "ADVERTISEMENT_DATE" : ISODate("2021-10-02T01:11:18.965+0000"),
...     "ADVERTISEMENT_MEDIA" : {
...         "MEDIA_NAME" : [
...             "FACEBOOK",
...             "TWITTER",
...             "TELEGRAM",
...             "INSTAGRAM"
...         ]
...     }
...
... })
WriteResult({ "nInserted" : 1 })
>
>
>
> db.ADVERTISEMENT.insert({
...     "ADVERTISEMENT_NO" : "A002",
...     "ADVERTISEMENT_DATE" : ISODate("2021-12-02T01:11:18.965+0000"),
...     "ADVERTISEMENT_MEDIA" : {
...         "MEDIA_NAME" : [
...             "FACEBOOK",
...             "TWITTER",
...             "TELEGRAM",
...             "INSTAGRAM"
...         ]
...     }
...
... })
WriteResult({ "nInserted" : 1 })
>
> db.ADVERTISEMENT.insert({
...     "ADVERTISEMENT_NO" : "A003",
...     "ADVERTISEMENT_DATE" : ISODate("2021-11-02T01:11:18.965+0000"),
...     "ADVERTISEMENT_MEDIA" : {
...         "MEDIA_NAME" : [
...             "FACEBOOK",
...             "TWITTER",
...             "TELEGRAM",
...             "INSTAGRAM"
...         ]
...     }
...
... })
WriteResult({ "nInserted" : 1 })
>
```

Performing the related quires

Outer joins :

The join that is performed here looks up the property number form advertisement and property collection for similarity of property number and it let joins the property number to the Branch collection.

```
> db.BRANCH.aggregate([{
... 
... $lookup:{
... 
...         from:"ADVERTISEMENT",
...         localField:"PROPERTY_NO",
...         foreignField:"PROPERTY_NO",
...         as:"PROPERTY"
... 
... },
... $lookup:{
... 
...     from:"PROPERTY",
...     localField:"PROPERTY_NO",
...     foreignField:"PROPRTY_NO",
...     as:"PROPERTY"
... }
... 
... }]).pretty()
{
        "_id" : ObjectId("62bd412994cb341370b26f70"),
        "BRANCH_NO" : "B001",
        "ADDRESS" : {
                "STREET_NO" : "00001",
                "STREET" : "JYATHA",
                "CITY" : "KATHMANDU",
                "PROVINCE" : "BAGMATI",
                "POSTAL_CODE" : "44600"
        },
        "BRANCH_MANAGER" : "SANDESH",
        "STAFF" : [
                {
                        "STAFF_NO" : "S001",
                        "LAST_NAME" : "SHARMA",
                        "FIRST_NAME" : "SITA",
                        "MIDDLE_NAME" : "DEVI",
                        "SEX" : "F",
                        "SALARY" : 5000
                },
```

```
1  db.BRANCH.aggregate([{
2  $lookup:{
3          from:"PROPERTY",
4          localField:"BRANCH_NO",
5          foreignField:"BRANCH_NO",
6          as:"PROPERTY_INFO"
7  }},
8  {   $unwind:{path:"$PROPERTY_INFO",preserveNullAndEmptyArrays: true }},
9  {
10         $project:{
11             "BRANCH_NO":1,
12             "BRANCH_MANAGER": 1,
13             "CITY":"GORKHA",
14             "COMPANY":"$PROPERTY_INFO.TENANT.BUSINESS_TENANT.BUSINESS_NAME"
15         }
16     }
17 ])
18
```

Raw Shell Output | Aggregate Query (line 1) ×

50 | Documents 1 to 4

BRANCH > BRANCH_NO

_id	BRANCH_NO	BRANCH_MANAGER	CITY	COMPANY
62bd412994cb3...	B001	SANDESH	GORKHA	
62bd422c94cb3...	B002	BIBEK	GORKHA	DBL CARE
62bd4f7694cb34...	B003	ANCHALA	GORKHA	RG INDUSTRIES
62bd53b394cb3...	B004	ALEX	GORKHA	

Temporal features

This temporal query projects the year and month of advertisements that were there.

```
> db.ADVERTISEMENT.find({ ADVERTISEMENT_DATE : { $gt:ISODate("2020-10-02T01:11:18.965+0000"),
... $lt:ISODate("2025-10-02T01:11:18.965+0000")}}).pretty()
{
        "_id" : ObjectId("62bd67eb94cb341370b26f78"),
        "ADVERTISEMENT_NO" : "A001",
        "ADVERTISEMENT_DATE" : ISODate("2021-10-02T01:11:18.965Z"),
        "ADVERTISEMENT_MEDIA" : {
                "MEDIA_NAME" : [
                        "FACEBOOK",
                        "TWITTER",
                        "TELEGRAM",
                        "INSTAGRAM"
                ]
        }
}
{
        "_id" : ObjectId("62bd680194cb341370b26f79"),
        "ADVERTISEMENT_NO" : "A002",
        "ADVERTISEMENT_DATE" : ISODate("2021-12-02T01:11:18.965Z"),
        "ADVERTISEMENT_MEDIA" : {
                "MEDIA_NAME" : [
                        "FACEBOOK",
                        "TWITTER",
                        "TELEGRAM",
                        "INSTAGRAM"
                ]
        }
}
{
        "_id" : ObjectId("62bd681c94cb341370b26f7a"),
        "ADVERTISEMENT_NO" : "A003",
        "ADVERTISEMENT_DATE" : ISODate("2021-11-02T01:11:18.965Z"),
        "ADVERTISEMENT_MEDIA" : {
                "MEDIA_NAME" : [
                        "FACEBOOK",
                        "TWITTER",
                        "TELEGRAM",
                        "INSTAGRAM"
                ]
        }
}
```

```
>
> db.ADVERTISEMENT.aggregate([{$project:{year:{$year:"$ADVERTISEMENT_DATE"},
... month:{$month:"$ADVERTISEMENT_DATE"}}}]).pretty()
{
        "_id" : ObjectId("62bd67eb94cb341370b26f78"),
        "year" : 2021,
        "month" : 10
}
{
        "_id" : ObjectId("62bd680194cb341370b26f79"),
        "year" : 2021,
        "month" : 12
}
{
        "_id" : ObjectId("62bd681c94cb341370b26f7a"),
        "year" : 2021,
        "month" : 11
}
>
```

ROLLUP

This roll up query sums up the salary of staff members who belong to a certain branch and groups them by the branch number.

```
db.BRANCH.aggregate( [

   {
         $unwind: "$STAFF" },
   {
      $group: {
        _id: "$BRANCH_NO",
        total_staff:{$sum:1},
        TOTAL_SALARY: { $sum: "$STAFF.SALARY" }
      }
    }
 ] ).pretty()
```

Raw Shell Output | Shell Output (Documents) ×

50 | Documents 1 to 4

Result > total_staff

_id	total_staff	TOTAL_SALARY
B003	3.0	500000.0
B004	3.0	1600000.0
B002	3.0	407000.0
B001	2.0	10000.0

Query a: A join of three or more tables –

The join that is performed here looks up the property number form advertisement and property collection for similarity of property number and it let joins the property number to the Branch collection in MongoDB. In sql it join the table of property_owner_type, Tenant and property where it look for the owner_no matching in property table and then Tenant_no matching in the lease_agreement table. The output will be the owner name who have the property on lease with its signing, starting and ending date.

SQL code

```
SELECT T.TENANT_NO,TREAT(VALUE(T) AS
PRIVATE_TENANT).FIRSTNAME
FIRSTNAME,TREAT(VALUE(T) AS
PRIVATE_TENANT).LASTNAME
LASTNAME,TREAT(VALUE(T) AS
BUSINESS_TENANT).BUSINESS_NAME
BUSINESS_NAME,LEASE_AGREEMENT_ID,P.PRO
PERTY_NO,P.PROPERTY_TYPE,SIGNING_DATE,
STARTING_DATE,ENDING_DATE
FROM TENANT T
INNER JOIN LEASE_AGREEMENT L
ON T.TENANT_NO = L.TENANT_NO
INNER JOIN PROPERTY P
ON P.PROPERTY_NO = L.PROPERTY_NO;
```

MongoDB code

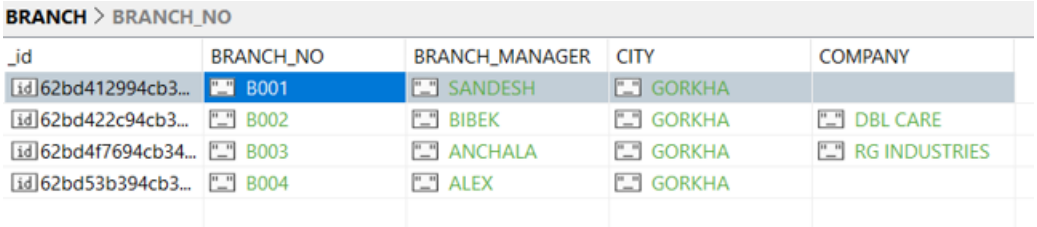
```
db.BRANCH.aggregate([{
$lookup:{
        from:"PROPERTY",
        localField:"BRANCH_NO",
        foreignField:"BRANCH_NO",
        as:"PROPERTY_INFO"
}},
{$unwind:{path:"$PROPERTY_INFO",
preserveNullAndEmptyArrays: true }},
    {
        $project:{
            "BRANCH_NO":1,
            "BRANCH_MANAGER":1,
            "CITY":"GORKHA",

"COMPANY":"$PROPERTY_INFO.TENANT.BUSINESS_TENANT.BUS
INESS_NAME"
    }
    }
```

Screenshots

SQL:

	TENANT_NO	FIRSTNAME	LASTNAME	BUSINESS_NAME	LEASE_AGREEMENT_ID	PROPERTY_NO	PROPERTY_TYPE	SIGNING_DATE	STARTING_DATE	ENDING_DATE
1	T006	(null)	(null)	COSMETIC	LA07	P006	COMMERCIAL	22-MAY-22 12.00.00.000000000 AM	22-MAY-22 12.00.00.000000000 AM	22-DEC-22 12.00.00.000000000 AM
2	T001	SANDESH	PANTA	(null)	LA01	P001	COMMERCIAL	20-FEB-20 12.00.00.000000000 AM	20-FEB-20 12.00.00.000000000 AM	20-FEB-21 12.00.00.000000000 AM
3	T002	SITA	SHARMA	(null)	LA02	P002	RESIDENT	20-MAR-20 12.00.00.000000000 AM	20-MAR-20 12.00.00.000000000 AM	20-MAR-21 12.00.00.000000000 AM
4	T003	BINO	PHIRI	(null)	LA03	P003	COMMERCIAL	20-APR-20 12.00.00.000000000 AM	20-APR-20 12.00.00.000000000 AM	20-APR-21 12.00.00.000000000 AM
5	T004	(null)	(null)	TULEK	LA0	P004	RESIDENT	20-APR-20 12.00.00.000000000 AM	20-APR-20 12.00.00.000000000 AM	20-APR-21 12.00.00.000000000 AM
6	T005	(null)	(null)	TULEK	LA05	P005	COMMERCIAL	20-APR-20 12.00.00.000000000 AM	20-APR-20 12.00.00.000000000 AM	20-APR-21 12.00.00.000000000 AM
7	T007	HARI	THAPA	(null)	LA06	P008	COMMERCIAL	20-APR-22 12.00.00.000000000 AM	20-APR-22 12.00.00.000000000 AM	20-APR-23 12.00.00.000000000 AM
8	T008	(null)	(null)	COSMETIC	LA08	P007	COMMERCIAL	18-JUN-22 12.00.00.000000000 AM	18-MAY-22 12.00.00.000000000 AM	18-DEC-22 12.00.00.000000000 AM

MONGODB:



Discussion:

The challenges faced during the implementation of join queries in MongoDB is that the implementation of the quires is much harder to execute, since it brings methods like aggregation and $lookup when trying to implement the quires so that proved to be challenging since it differs from sql implementation. On the other hand, thou it proved to be challenging to implement the quires, the query processing was much more

efficient and faster as compared to sql because MongoDB uses the database wide locking to allow only one write query per database at a time (SuvithaVani, 2020). Rajat et al (R. Aghi, 2015) compared SQL and MongoDB and find that MongoDB works better for efficient complex queries of join with large dataset. Complex queries involving multiple joins I have done in Mongodb and Oracle, MongoDb performed better than the oracle because of its data structure allowing it to accommodate any type of data. The database sql was familiar to me since last two and throught these course of ADT which make SQL little bit more easy while implementating join queries. Whereas MongoDB was little bit of confusing because of using the terms like pipeline, $lookup and aggregation.

In MongoDB, joining two documents is likewise not straightforward. MongoDB does allow left outer joins (lookup), however it is still in its inception (Hecht, 2011). It might not be achievable if it calls for combining data from various collections into a single query. I must thus perform several queries, which may make my code appear a little disorganized.

Query b: A query which requires use of either a nested table or subtypes

This temporal query projects the year and month of advertisements that were there in MongoDB. This query will display an agreement that was made and entered in the last 35-day equivalent to 5weeks. retrieving the documents that match the nested fields in MongoDB.

SQL code	MongoDB code
SELECT p.OWNER_NO, TREAT(VALUE(p) AS BUSINESS_OWNER).BUSINESS_OWNER_NAME,CITY FROM PROPERTY_OWNER_ObjType p WHERE VALUE(p) IS OF (BUSINESS_OWNER);	db.PROPERTY.find({ "LEASE_AGREEMENT.LEASE_AGREEMENT_ID": "LA01", "TENANT.TENANT_NO":"T001",}).pretty()
Screenshots	

SQL:

	OWNER_NO	TREAT(VALUE(P)ASBUSINESS_OWNER).BUSINESS_OWNER_NAME	CITY
1	0015	RG INDUSTRIES	BHAKTAPUR

MongoDB:

PROPERTY > PROPERTY_NO

_id	PROPERTY_NO	PROPERTY_TYPE	PROPERTY_OWNER	LEASE_AGREEMENT	TENANT	BRANCH_NO
62bd57b794cb3...	P001	RESIDENTAL	{ 7 fields }	{ 4 fields }	{ 2 fields }	B001

Discussion:

The benefit of using a document database is that MongoDB may immediately store your object as a single document. An ORM is not required in this case. The MongoDB document might look awful if your data model allows objects to have recursive children, which means the same object type can be a child of another object 'n' times (SuvithaVani, 2020). It might be challenging to index, search, and organize these embedded recursive documents. NoSQL doesn't support any relations between different data types. In some ways you

structure your database according to the result you want to see on screen rather than trying to put it in some normalized format. MongoDB makes it simple to insert one document within another. Field-value pairs are enclosed behind curly brackets ({}) when representing documents in the mongo shell (R. Aghi, 2015). Curly braces may now be used to embed an other document inside of these fields, and this document may include field-value pairs or another sub-document. Creating a nested table in SQL was confusing because we have to create a table that is embedded within another table and also creating type as an object, table of object type, nested table and inserting data into it. I find that using MongoDB on the query was easy rather than SQL.

Query c: A query using temporal features (e.g., timestamps, intervals, etc.) of Oracle SQL

Provide a description of your query here

SQL code

```
SELECT TENANT_NO, PROPERTY_NO,
LEASE_AGREEMENT_ID,SIGNING_DATE,STARTING_DATE,ENDIN
G_DATE
FROM LEASE_AGREEMENT
VERSIONS PERIOD FOR LEASE_HIST_TIME BETWEEN
TRUNC(SYSDATE)-35 AND TRUNC(SYSDATE);
```

MongoDB code

```
db.ADVERTISEMENT.find({
ADVERTISEMENT_DATE : {
$gt:ISODate("2020-10-
02T01:11:18.965+0000"),
$lt:ISODate("2025-10-
02T01:11:18.965+0000")}}).pretty()

db.ADVERTISEMENT.aggregate([{$project:{
year:{$year:"$ADVERTISEMENT_DATE"},
month:{$month:"$ADVERTISEMENT_DATE"}}}]
).pretty()
```

Screenshots:

SQL:

	TENANT_NO	PROPERTY_NO	LEASE_AGREEMENT_ID	SIGNING_DATE	STARTING_DATE	ENDING_DATE
1	T007	P008	LA06	20-APR-22 12.00.00.000000000 AM	20-APR-22 12.00.00.000000000 AM	20-APR-23 12.00.00.000000000 AM
2	T006	P006	LA07	22-MAY-22 12.00.00.000000000 AM	22-MAY-22 12.00.00.000000000 AM	22-DEC-22 12.00.00.000000000 AM
3	T008	P007	LA08	18-JUN-22 12.00.00.000000000 AM	18-MAY-22 12.00.00.000000000 AM	18-DEC-22 12.00.00.000000000 AM

MongoDB:

_id	ADVERTISEMENT_NO	ADVERTISEMENT_D...	ADVERTISEMENT_MEDIA
62bd67eb94cb3...	A001	2021-10-02T01:...	{ 1 fields }
62bd680194cb3...	A002	2021-12-02T01:...	{ 1 fields }
62bd681c94cb3...	A003	2021-11-02T01:...	{ 1 fields }

Discussion:

Timestamp is a data type and function in the Standard Structured Query Language (SQL) that enables us to store and manipulate date and time data values, often without defined time zones (Pedamkar, 2017). I have find that the timestamp queries in sql was very hard to convert to mongodb type language and cannot find out what's the equivalent date_trunc.

Firstly there was error output while trying to execute the query in mongodb and after digging sometime I find out that I was missing quotes around the format field. I also find out that I don't need to have a "null"

id. It was little bit of easy to perform these query in sql because we were previously taught sql in 2 years classes and I was familiar with the sql query. Whereas mongoDB was new to me so, that why I have to do little bit of research on temporal features in mongodb which consume my time through out the session.

Query d: A query using OLAP (e.g., ROLLUP, CUBE, PARTITION) features of Oracle SQL

The roll up query aggregates the grand totals and groups them using the staff number and branch number in MonogDB. This roll up query sums up the salary of staff members who belong to a certain branch and groups them by the branch number in SQL.

SQL code

```
SELECT STAFF_NO,BRANCH_NO, SUM(SALARY) AS
TOTAL_SALARY
FROM STAFF
GROUP BY ROLLUP(STAFF_NO, BRANCH_NO)
ORDER BY STAFF_NO, BRANCH_NO;
```

MongoDB code

```
db.BRANCH.aggregate( [

   {
         $unwind: "$STAFF" },
   {
      $group: {
        _id: "$BRANCH_NO",
        total_staff:{$sum:1},
        TOTAL_SALARY: { $sum: "$STAFF.SALARY"
}
      }
    }
 ] ).pretty()
```

Screenshots

SQL:

	STAFF_NO	BRANCH_NO	TOTAL_SALARY
1	S001	B0001	5000
2	S001	(null)	5000
3	S002	B0002	7000
4	S002	(null)	7000
5	S003	B0003	10000
6	S003	(null)	10000
7	S004	B0003	1000
8	S004	(null)	1000
9	S005	B0002	20000
10	S005	(null)	20000
11	S007	B0003	50000

MongoDB:

_id	total_staff	TOTAL_SALARY
B003	3.0	500000.0
B004	3.0	1600000.0
B002	3.0	407000.0
B001	2.0	10000.0

Discussion:

Writing queries in Mongo is tedious and exceedingly frustrating because of the extra characters that must be typed, such as quotes, brackets, square brackets, and colons. Sometimes it takes a lot of typing to get any kind of comprehensible result, when SQL would simply need a few short lines and appear more nicer.

Individual records in MongoDB are kept as documents, which are collections of fields with a dynamic structure. Because each collection does not have to have the same set of fields, it is more versatile than RDBMS.

Records in SQL databases are kept in rows inside a table, limiting dynamic categorization and storing of hierarchical data. However, SQL Relational data may be matched using common properties in a simplified manner, which might be advantageous depending on the use case (Vrbsky, 2013).

References

Hecht, R. a. J. S., 2011. NOSQL Evaluation A USE Case Orineted Survey. *Proceedings International Conference on Cloud and Service Computin,* pp. 12-14.

Pedamkar, P., 2017. SQL Timestamp.

R. Aghi, S. M. R. C. S. C. a. N. B., 2015. A comprehensive comparison of SQL and MongoDB databases. *International Journal of Scientific and Research Publications,* Volume 5, pp. 325-330.

SuvithaVani, P. &., 2020. A survey on RDBMS and NoSQL Databases. *International Conference on Computer Communication and Informatics,* 5(4), pp. 1-7.

Vrbsky, S., 2013. Comparing nosql mongodb to an sql db. *Comparing nosql mongodb to an sql db.*