Table of Contents

Star Schema 3

Exporting the data from a Microsoft Access database into Oracle 4

Creating the Staging Area 6

Creating and Populating the tables 7

- **Time_Dim** 7
- **Location_Dim** 8
- **Measurement_Dim** 9
- **Fact_Table** 11

Data Cleansing 12

QUERIES 14

Python code used for connecting to Oracle 15

- **Installing the cx_Oracle package** 15
- **Importing the libraries and setting the Oracle Client address** 15
- **Creating a connection to Oracle** 15
- **Establishing the connection** 15
- **Importing preprocessing from sklearn** 15
- **Splitting the data** 15
- **Fitting KNN Algorithm model into the training set** 16
- **Performing Cross-Validation** 16
- **Finding the optimal number of K neighbors** 17

Evaluation 17

- **Classification Report** 17
- **Accuracy Score** 17
- **Confusion Matrix** 18

Summary 18

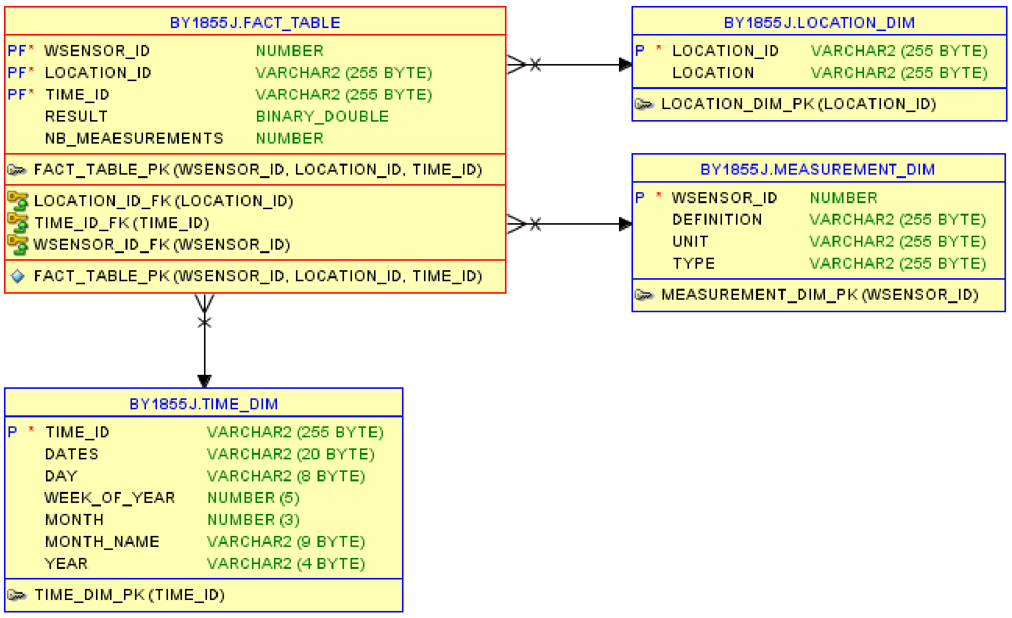
Star Schema

Star Schema consists of 3 different dimensions: Location Dimension, Time Dimension and Measurement Dimension. For Water Quality monitoring, water samples are taken periodically and data are provided from certain sampling points. However, since this method is not economical, the samplesamplingPointnotation column has added to the star schema as Location and the samplesamplingPointlabel column has added as Location_ID in order to accurately predict the trend of the data and the future parameter value.

Another dimension is the Measurement dimension, which includes the sensor field, which is a decisive factor for predicting the future parameter values of the data appropriately. Each parameter requires a specific sensor for testing, hence respectively wsensor_id (determinandnotation), type(Determinandlabel),Definition (determinanddefinition) fields have been added to the star schema.

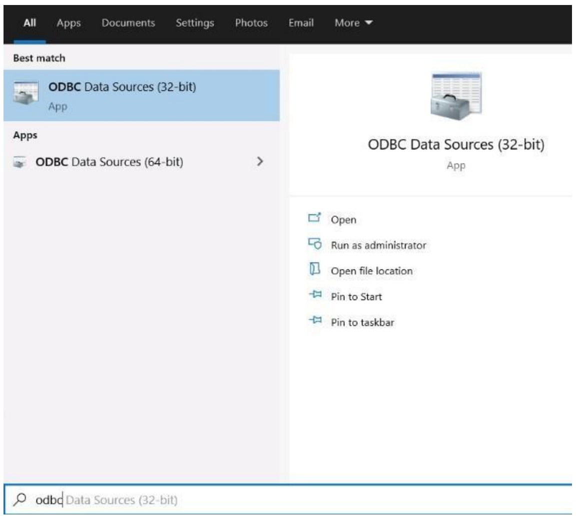
In order to make logical measurements, date, time, week, month and year information of the samples taken from certain points is needed, hence the Time Dimension had been added to the star schema. Time ensures consistency across columns for accurate measurement.

Fact table stores quantitative information from dimension tables and also includes results and nb_measurements fields in order to monitor and analyse the water quality with the results obtained according to time, sensor and location.

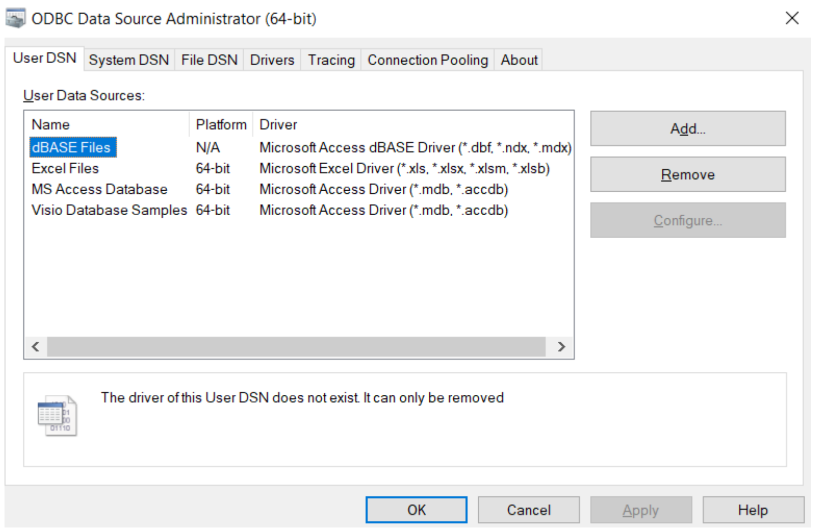

BY1855J.FACT_TABLE
PF* WSENSOR_ID NUMBER
PF* LOCATION_ID VARCHAR2 (255 BYTE)
PF* TIME_ID VARCHAR2 (255 BYTE)
RESULT BINARY_DOUBLE
NB_MEAESUREMENTS NUMBER
FACT_TABLE_PK (WSENSOR_ID, LOCATION_ID, TIME_ID)
LOCATION_ID_FK (LOCATION_ID)
TIME_ID_FK (TIME_ID)
WSENSOR_ID_FK (WSENSOR_ID)
FACT_TABLE_PK (WSENSOR_ID, LOCATION_ID, TIME_ID)
BY1855J.LOCATION_DIM
P * LOCATION_ID VARCHAR2 (255 BYTE)
LOCATION VARCHAR2 (255 BYTE)
LOCATION_DIM_PK (LOCATION_ID)
BY1855J.MEASUREMENT_DIM
P * WSENSOR_ID NUMBER
DEFINITION VARCHAR2 (255 BYTE)
UNIT VARCHAR2 (255 BYTE)
TYPE VARCHAR2 (255 BYTE)
MEASUREMENT_DIM_PK (WSENSOR_ID)
BY1855J.TIME_DIM
P * TIME_ID VARCHAR2 (255 BYTE)
DATES VARCHAR2 (20 BYTE)
DAY VARCHAR2 (8 BYTE)
WEEK_OF_YEAR NUMBER (5)
MONTH NUMBER (3)
MONTH_NAME VARCHAR2 (9 BYTE)
YEAR VARCHAR2 (4 BYTE)
TIME_DIM_PK (TIME_ID)

Exporting the data from a Microsoft Access database into Oracle

- Type ODBC Data Sources 64-bit on the start menu



- Click Add button to create a new data source under USER DSN tab

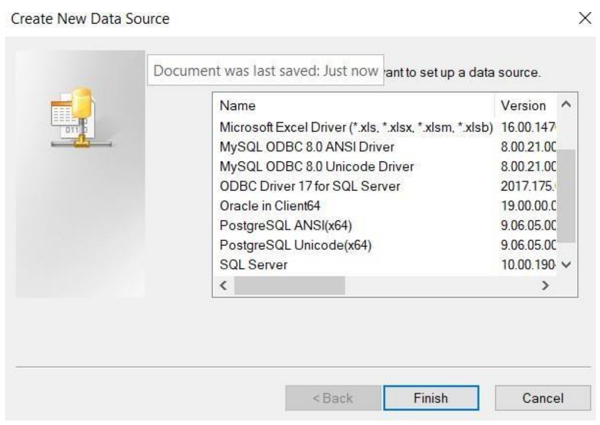


User Data Sources:

Name	Platform	Driver
dBASE Files	N/A	Microsoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)
Excel Files	64-bit	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)
MS Access Database	64-bit	Microsoft Access Driver (*.mdb, *.accdb)
Visio Database Samples	64-bit	Microsoft Access Driver (*.mdb, *.accdb)

The driver of this User DSN does not exist. It can only be removed

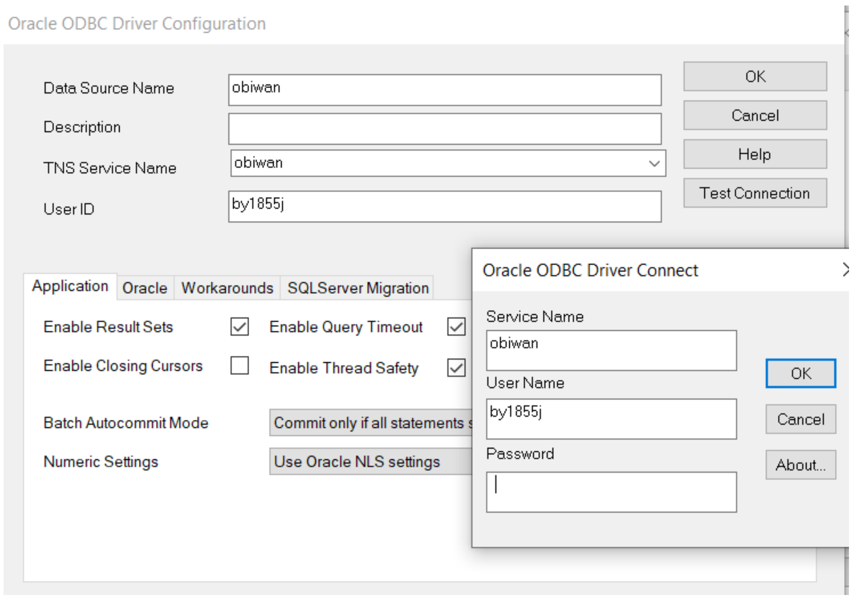
- Select " Oracle in Client64" on the popup that appears and click next.



...ant to set up a data source.

Name	Version
Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)	16.00.147
MySQL ODBC 8.0 ANSI Driver	8.00.21.0C
MySQL ODBC 8.0 Unicode Driver	8.00.21.0C
ODBC Driver 17 for SQL Server	2017.175.
Oracle in Client64	19.00.00.C
PostgreSQL ANSI(x64)	9.06.05.0C
PostgreSQL Unicode(x64)	9.06.05.0C
SQL Server	10.00.190

- Give the DSN a name, select the TNS Service name from the list and enter the userid. Click test connection. Enter your password.


Oracle ODBC Driver Configuration

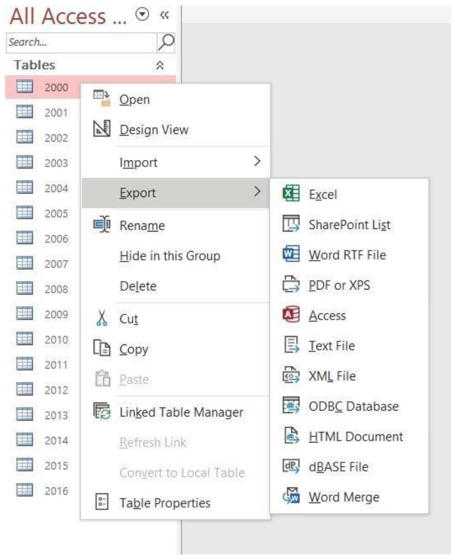
Data Source Name	obiwan
Description	
TNS Service Name	obiwan
User ID	by1855j

Enable Result Sets ✓ Enable Query Timeout ✓
Enable Closing Cursors ☐ Enable Thread Safety ✓
Batch Autocommit Mode: Commit only if all statements s
Numeric Settings: Use Oracle NLS settings

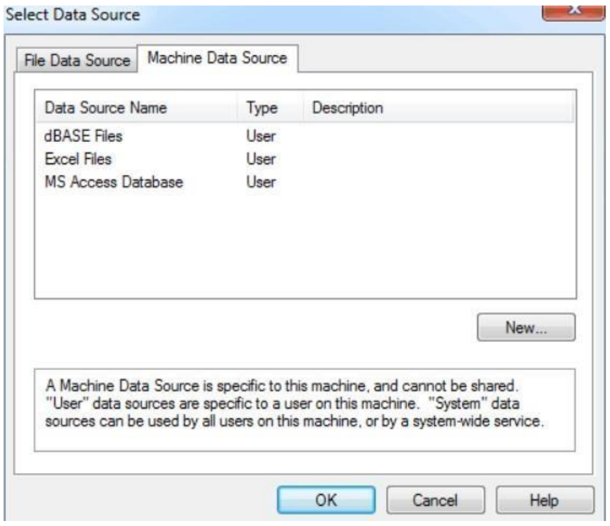
Oracle ODBC Driver Connect

Service Name	obiwan
User Name	by1855j
Password	

- Go back to Microsoft Access database. Right-click and select export. Choose ODBC Database.



- Click Ok button.
- Click the Machine Data Source Tab and choose obiwan Data source Name and the Type user. Then click the OK button.



Data Source Name	Type	Description
dBASE Files	User	
Excel Files	User	
MS Access Database	User	

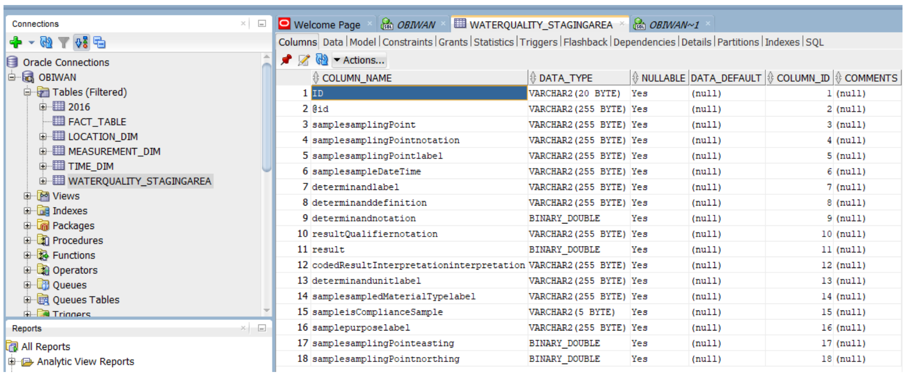
A Machine Data Source is specific to this machine, and cannot be shared. "User" data sources are specific to a user on this machine. "System" data sources can be used by all users on this machine, or by a system-wide service.

- Enter the password for the Oracle Driver Connect popup.
- Click the close button.
- Go to SQL developer and Click Refresh button. The exported data appears on the screen.

Creating the Staging Area

To create the staging area, a copy of the table 2000 was made and the copied table was named as WATERQUALITY_STAGINGAREA. The data of all tables from 2001 to 2016 were transferred to the WATERQUALITY_STAGINGAREA table with the following code.

```
Insert Into WATERQUALITY_STAGINGAREA

Select*From 2001;

Insert Into WATERQUALITY_STAGINGAREA

Select*From 2002;

  .

  .

  .

Insert Into WATERQUALITY_STAGINGAREA

Select*From 2016;
```



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	VARCHAR2 (20 BYTE)	Yes	(null)	1	(null)
2	@id	VARCHAR2 (255 BYTE)	Yes	(null)	2	(null)
3	samplesamplingPoint	VARCHAR2 (255 BYTE)	Yes	(null)	3	(null)
4	samplesamplingPointnotation	VARCHAR2 (255 BYTE)	Yes	(null)	4	(null)
5	samplesamplingPointlabel	VARCHAR2 (255 BYTE)	Yes	(null)	5	(null)
6	samplesampleDateTime	VARCHAR2 (255 BYTE)	Yes	(null)	6	(null)
7	determinandlabel	VARCHAR2 (255 BYTE)	Yes	(null)	7	(null)
8	determinanddefinition	VARCHAR2 (255 BYTE)	Yes	(null)	8	(null)
9	determinandnotation	BINARY_DOUBLE	Yes	(null)	9	(null)
10	resultQualifiernotation	VARCHAR2 (255 BYTE)	Yes	(null)	10	(null)
11	result	BINARY_DOUBLE	Yes	(null)	11	(null)
12	codedResultInterpretationinterpretation	VARCHAR2 (255 BYTE)	Yes	(null)	12	(null)
13	determinandunitlabel	VARCHAR2 (255 BYTE)	Yes	(null)	13	(null)
14	samplesampledMaterialTypelabel	VARCHAR2 (255 BYTE)	Yes	(null)	14	(null)
15	sampleisComplianceSample	VARCHAR2 (5 BYTE)	Yes	(null)	15	(null)
16	samplepurposelabel	VARCHAR2 (255 BYTE)	Yes	(null)	16	(null)
17	samplesamplingPointeasting	BINARY_DOUBLE	Yes	(null)	17	(null)
18	samplesamplingPointnorthing	BINARY_DOUBLE	Yes	(null)	18	(null)

Creating and Populating the tables

Time_Dim

Time_Dim table was created as below. And link to the fact table to find out the number of sensor measurements collected by type of sensor by week or year , the number of measurements made by location by month , the average number of measurements covered for PH by year etc. time_id has chosen as the primary key since the values in the time_id field are already unique. When the time table was first built, a new primary key was defined by generating a sequence, instead of utilising time_id but it was quickly realised that this was unnecessary because time_id performed the same function. The code of sequence is also included below. According to the task that the field performs in the table, data types and lengths has defined as follows.

This table was populated with the data in the samplesampleDateTime field of the WATERQUALITY_STAGINGAREA table by using cursor. The samplesampleDateTime field contains the date and time values together. Using cursor to fill time_id, dates, day, week_of_year, month, month name and year fields, correct formats was obtained with to_number, to_char ,substr functions and data was filled into the table.

Creating Table:

```
Create table Time_Dim(

TIME_ID VARCHAR2(255 BYTE),

primary key(TIME_ID),

Dates  varchar(20),

Day varchar(8),

Week_of_Year number(5,0),

Month number(3,0),

Month_Name varchar(9),

Year varchar(4));
```

Creating Sequence(deleted) :

```
create sequence Time_Dim_seq;

select Time_Dim_seq.nextval from dual;

Create or replace trigger Time_Dim_Pk

before insert on Time_Dim

 for each row

  begin

   select Time_Dim_seq.nextval into :new.TIME_ID from dual;

  end ;

/
```

Populating the Table:

```
declare

Cursor ws is

select distinct "samplesampleDateTime" as d from waterquality_stagingarea;

begin

for w in ws loop

insert into Time_Dim values(w.d,

TO_CHAR(to_date(substr(w.d,0,10), 'YYYY.MM.DD')),

TO_CHAR(to_date(substr(w.d,0,10), 'YYYY.MM.DD'), 'DD'),

to_number(TO_CHAR(to_date(substr(w.d,0,10), 'YYYY.MM.DD'), 'IW')),

to_number(TO_CHAR(to_date(substr(w.d,0,10), 'YYYY.MM.DD'), 'MM')),

TO_CHAR(to_date(substr(w.d,0,10), 'YYYY.MM.DD'), 'Month'),

TO_CHAR(to_date(substr(w.d,0,10), 'YYYY.MM.DD'), 'YYYY'));

end loop;

end;

/
```

Location_Dim

The Location_Dim table was created as follows. Location_ID was defined as primary key and this field corresponds to samplesamplingPointnotation field. Since all values in this field are unique, it was set as the primary key. The Location field gives the definitions of all locations. The data types of the fields were determined as follows in order to populate the data in the fact table in the most appropriate way and to ensure consistency between the tables.

Insert Into Select command was used as below to populate the table. The Select Distinct command was used to avoid repetitive records while the data was being retrieved from the WATERQUALITY_STAGINGAREA table.

Creating Table:

```
create table Location_Dim(

Location_ID VARCHAR2(255 BYTE),

Location VARCHAR2(255 BYTE),

Primary Key(Location_ID));
```

Populating Location Dimension:

```
INSERT INTO Location_Dim (Location_ID, Location)

SELECT distinct "samplesamplingPointnotation", "samplesamplingPointlabel"

FROM waterquality_stagingarea;
```

Measurement_Dim

Measurement_Dim table was created as follows. WSensor_ID was defined as primary key and this field corresponds to determinandnotationfield. Since all values in this field are unique, it was set as the primary key. Definition (determinanddefinition), type(Determinandlabel),Unit(determinandunitlabel) fields were created with the following data types. Data types were defined in the most appropriate way to facilitate the connection between tables.

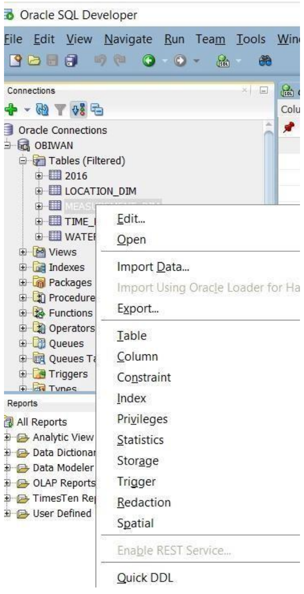
Sequence creation was abandoned for the same reason as the time_dim table. The sequence created for this table was defined by the following method.

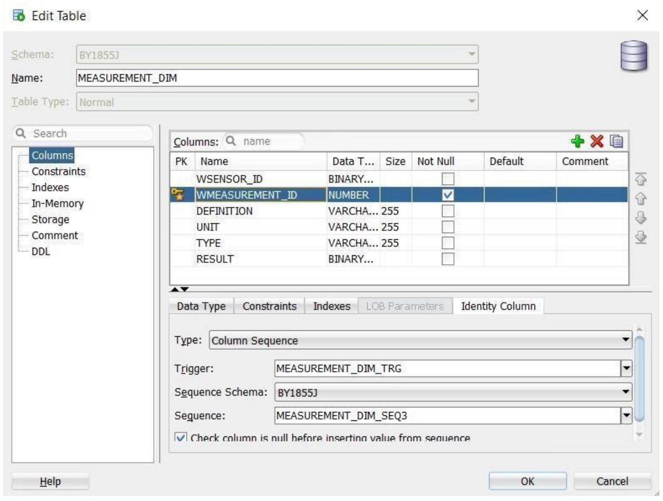
Insert Into Select command was used as below to populate the table. The Select Distinct command was used to avoid repetitive records while the data was being retrieved from the WATERQUALITY_STAGINGAREA table.

Creating Table:

```
create table Measurement_Dim(

WSensor_ID number,

Definition VARCHAR2(255 BYTE),

Unit varchar2(255 byte),

Type VARCHAR2(255 BYTE),

Primary Key(WSensor_ID));
```

Creating Sequence (deleted):



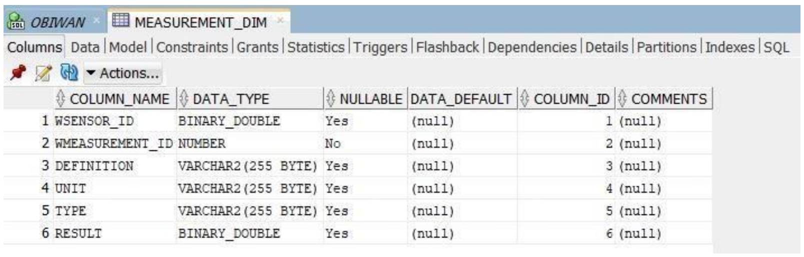


Schema: BY1855J
Name: MEASUREMENT_DIM
Table Type: Normal

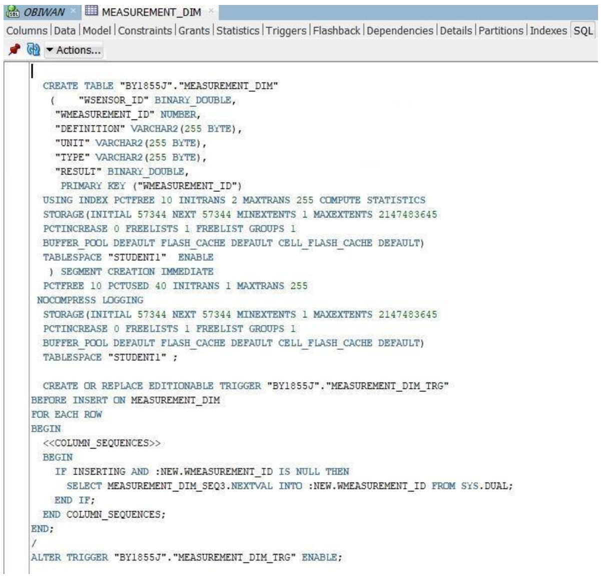
Columns:

PK	Name	Data T...	Size	Not Null	Default	Comment
	WSENSOR_ID	BINARY...		☐		
🔑	WMEASUREMENT_ID	NUMBER		✓		
	DEFINITION	VARCHA...	255	☐		
	UNIT	VARCHA...	255	☐		
	TYPE	VARCHA...	255	☐		
	RESULT	BINARY...		☐		

Type: Column Sequence
Trigger: MEASUREMENT_DIM_TRG
Sequence Schema: BY1855J
Sequence: MEASUREMENT_DIM_SEQ3
✓ Check column is null before inserting value from sequence


OBIWAN
MEASUREMENT_DIM
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	WSENSOR_ID	BINARY_DOUBLE	Yes	(null)	1	(null)
2	WMEASUREMENT_ID	NUMBER	No	(null)	2	(null)
3	DEFINITION	VARCHAR2(255 BYTE)	Yes	(null)	3	(null)
4	UNIT	VARCHAR2(255 BYTE)	Yes	(null)	4	(null)
5	TYPE	VARCHAR2(255 BYTE)	Yes	(null)	5	(null)
6	RESULT	BINARY_DOUBLE	Yes	(null)	6	(null)


OBIWAN
MEASUREMENT_DIM
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Actions...

```
  CREATE TABLE "BY1855J"."MEASUREMENT_DIM"
   (    "WSENSOR_ID" BINARY_DOUBLE,
    "WMEASUREMENT_ID" NUMBER,
    "DEFINITION" VARCHAR2(255 BYTE),
    "UNIT" VARCHAR2(255 BYTE),
    "TYPE" VARCHAR2(255 BYTE),
    "RESULT" BINARY_DOUBLE,
     PRIMARY KEY ("WMEASUREMENT_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 57344 NEXT 57344 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "STUDENT1"  ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 57344 NEXT 57344 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "STUDENT1" ;

  CREATE OR REPLACE EDITIONABLE TRIGGER "BY1855J"."MEASUREMENT_DIM_TRG"
BEFORE INSERT ON MEASUREMENT_DIM
FOR EACH ROW
BEGIN
  <<COLUMN_SEQUENCES>>
  BEGIN
    IF INSERTING AND :NEW.WMEASUREMENT_ID IS NULL THEN
      SELECT MEASUREMENT_DIM_SEQ3.NEXTVAL INTO :NEW.WMEASUREMENT_ID FROM SYS.DUAL;
    END IF;
  END COLUMN_SEQUENCES;
END;
/
ALTER TRIGGER "BY1855J"."MEASUREMENT_DIM_TRG" ENABLE;
```

Populating Measurement Dimension:

```
INSERT INTO measurement_dim(wsensor_id, Definition,Unit, Type)

SELECT distinct
"determinandnotation","determinanddefinition","determinandunitlabel","determinandlabel"

FROM waterquality_stagingarea
```

Fact_Table

Fact_Table was created as follows. Foreign Keys enables to easily find tables relation and enforce data consistency and validate fact table has no other data than of dimension table. Foreign keys were created with specific constraint name as below. Primary key was created as Fact_Table_PK. Because there is more than one column, composite primary key was used to specify the primary key of table.

Data types were defined in the most appropriate way to facilitate the connection between tables.

"Delete Cascade" was used for deleting the referencing rows in the fact table when the referenced row is deleted in the dimension tables which have a primary key.

The table was populated with the data from location_dim, time_dim, waterquality_stagingarea, measurement_dim tables by using cursor.

NB_Meaesurements was created to show number of measurements by WSENSOR_ID,Location_ID, TIME_ID as an indicator by using where condition.

Creating Table:

```
CREATE TABLE Fact_Table

(WSensor_ID number,

Location_ID VARCHAR2(255 BYTE),

TIME_ID VARCHAR2(255 BYTE),

Result BINARY_DOUBLE,

NB_Meaesurements number,

CONSTRAINT WSensor_ID_FK FOREIGN KEY (WSensor_ID) REFERENCES
Measurement_Dim(WSensor_ID) ON DELETE CASCADE,

CONSTRAINT Location_ID_FK FOREIGN KEY (Location_ID) REFERENCES location_Dim(Location_ID)
ON DELETE CASCADE,

CONSTRAINT TIME_ID_FK FOREIGN KEY (TIME_ID) REFERENCES TIME_DIM(TIME_ID) ON DELETE
CASCADE,

CONSTRAINT Fact_Table_PK PRIMARY KEY (WSensor_ID, Location_ID, TIME_ID));
```

Populating Fact_Table:

```
DECLARE

CURSOR factData IS

SELECt distinct md. WSENSOR_ID,ld.Location_ID,td.TIME_ID,ws."result",

COUNT(*) AS NB_Meaesurements

from measurement_dim md,location_dim ld,time_dim td,waterquality_stagingarea ws

where ws."determinandnotation"= md.wsensor_id AND
ws."samplesamplingPointnotation"=ld.location_id and ws."samplesampleDateTime" = td.time_id
```

```
GROUP BY md.WSENSOR_ID,ld.Location_ID,td.TIME_ID,ws."result";

BEGIN

FOR fd IN factData

LOOP

   INSERT INTO fact_table
VALUES(fd.WSensor_ID,fd.Location_ID,fd.TIME_ID,fd."result",fd.NB_Meaesurements);

END LOOP;

END;

/
```

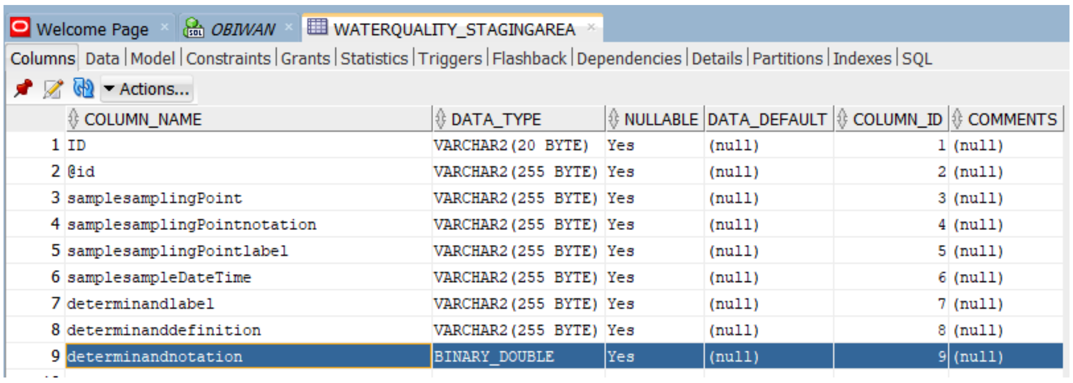
Data Cleansing

- Only a few data cleaning operations were performed on the staging area, since the implementation of the data cleaning step on the star schema dimensions used saves time and erroneous data more visible. Before the data were used by the dimensions, the unsampled determinandlabel values were deleted using the code below.

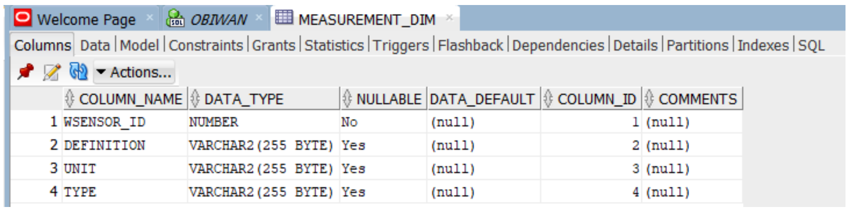
  ```
  delete from WATERQUALITY_STAGINGAREA where "determinandlabel" = 'NO FLOW/SAMP';
  ```

- Only the following columns, which were required for 3 different dimensions in the star schema, had been transferred from the staging area: samplesamplingPointnotation, samplesamplingPointlabel, samplesampleDateTime, determinandlabel, determinanddefinition, determinandnotation, result, determinandunitlabel.

- The data of the wsensor_id field in the measurement_dim table was taken from the determinandnotation field in the WATERQUALITY_STAGINGAREA table. The data type of the WSENSOR_ID field was set to number, not binary_double, unlike the determinandnotation field. The WSENSOR_ID field was set as the primary key for the measurement_dim table, hence this approach was used to remove values after the comma.

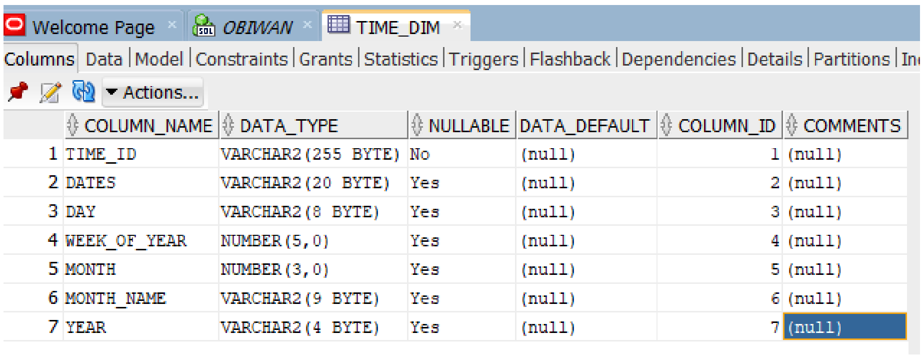

Welcome Page
OBIWAN
WATERQUALITY_STAGINGAREA
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	VARCHAR2(20 BYTE)	Yes	(null)	1	(null)
2	@id	VARCHAR2(255 BYTE)	Yes	(null)	2	(null)
3	samplesamplingPoint	VARCHAR2(255 BYTE)	Yes	(null)	3	(null)
4	samplesamplingPointnotation	VARCHAR2(255 BYTE)	Yes	(null)	4	(null)
5	samplesamplingPointlabel	VARCHAR2(255 BYTE)	Yes	(null)	5	(null)
6	samplesampleDateTime	VARCHAR2(255 BYTE)	Yes	(null)	6	(null)
7	determinandlabel	VARCHAR2(255 BYTE)	Yes	(null)	7	(null)
8	determinanddefinition	VARCHAR2(255 BYTE)	Yes	(null)	8	(null)
9	determinandnotation	BINARY_DOUBLE	Yes	(null)	9	(null)


Welcome Page
OBIWAN
MEASUREMENT_DIM
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	WSENSOR_ID	NUMBER	No	(null)	1	(null)
2	DEFINITION	VARCHAR2(255 BYTE)	Yes	(null)	2	(null)
3	UNIT	VARCHAR2(255 BYTE)	Yes	(null)	3	(null)
4	TYPE	VARCHAR2(255 BYTE)	Yes	(null)	4	(null)

- The samplesampleDateTime field in WATERQUALITY_STAGINGAREA was used as the basis to create all the fields in the time_dim table. The samplesampleDateTime field of type VARCHAR2(255 BYTE) is in the format containing the date and time at the same time. In order to properly populate the data and improve performance, the data types of the columns in the time_dim table had been changed as in the figure below. The sizes of all fields had been determined as they should be.
  In order to populate the data into 7 fields in the time_dim table, it was extracted from the samplesampleDateTime field using the TO_CHAR, to_date,substr, to_number functions as in the code below.


Welcome Page
OBIWAN
TIME_DIM
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | In
Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	TIME_ID	VARCHAR2(255 BYTE)	No	(null)	1	(null)
2	DATES	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3	DAY	VARCHAR2(8 BYTE)	Yes	(null)	3	(null)
4	WEEK_OF_YEAR	NUMBER(5,0)	Yes	(null)	4	(null)
5	MONTH	NUMBER(3,0)	Yes	(null)	5	(null)
6	MONTH_NAME	VARCHAR2(9 BYTE)	Yes	(null)	6	(null)
7	YEAR	VARCHAR2(4 BYTE)	Yes	(null)	7	(null)

```
declare

Cursor ws is

select distinct "samplesampleDateTime" as d from waterquality_stagingarea;

begin

for w in ws loop

insert into Time_Dim values(w.d,

TO_CHAR(to_date(substr(w.d,0,10), 'YYYY.MM.DD')),

TO_CHAR(to_date(substr(w.d,0,10), 'YYYY.MM.DD'), 'DD'),

to_number(TO_CHAR(to_date(substr(w.d,0,10), 'YYYY.MM.DD'), 'IW')),

to_number(TO_CHAR(to_date(substr(w.d,0,10), 'YYYY.MM.DD'), 'MM')),

TO_CHAR(to_date(substr(w.d,0,10), 'YYYY.MM.DD'), 'Month'),

TO_CHAR(to_date(substr(w.d,0,10), 'YYYY.MM.DD'), 'YYYY'));

end loop;

end;

/
```

- To avoid duplicate data, the select*distinct command was used while populating the data for all 3 dimensions.

```
select distinct "samplesampleDateTime" as d fromwaterquality_stagingarea;
```

```
SELECT distinct "samplesamplingPointnotation", "samplesamplingPointlabel" from
waterquality_stagingarea;

SELECT distinct
"determinandnotation","determinanddefinition","determinandunitlabel","determinandlabel"

FROM waterquality_stagingarea;
```

QUERIES

The list of water sensors measured by type of it by month ;

```
select md.WSENSOR_ID, md."DEFINITION", td.month from measurement_dim md , time_dim td,
FACT_TABLE fd

where md.WSENSOR_ID=fd.WSENSOR_ID and td.time_id=fd.time_id ;
```

The number of sensor measurements collected by type of sensor by week ;

```
select md."DEFINITION", td.WEEK_OF_YEAR,count(fd.result) as NB_measurement_Sensor_Week
from measurement_dim md , time_dim td, FACT_TABLE fd

where md.WSENSOR_ID=fd.WSENSOR_ID and td.time_id=fd.time_id

group by(md."DEFINITION",td.WEEK_OF_YEAR);
```

The number of measurements made by location by month ;

```
select ld.location, td.month,count(fd.result) as NB_measurement_Location_Month from
location_dim ld , time_dim td, FACT_TABLE fd

where ld.location_id=fd.location_id and td.time_id=fd.time_id

group by(ld.location,td.month);
```

The average number of measurements covered for PH by year ;

```
select md."DEFINITION", td.year,avg(fd.result) as AVG_measurement_PH_Year from
measurement_dim md , time_dim td, FACT_TABLE fd

where md.WSENSOR_ID=fd.WSENSOR_ID and td.time_id=fd.time_id and md.WSENSOR_ID=61

group by(md."DEFINITION",td.year);
```

The average value of Nitrate measurements by locations by year;

```
select md."DEFINITION", ld.location, td.year,avg(fd.result) as
AVG_measurement_Nitrate_Location_Year from measurement_dim md,location_dim ld , time_dim
td, FACT_TABLE fd

where md.WSENSOR_ID=fd.WSENSOR_ID and ld.location_id=fd.location_id and
td.time_id=fd.time_id and md.WSENSOR_ID=117
```

```
group by(md."DEFINITION", ld.location,td.year);
```

Python code used for connecting to Oracle

Installing the cx_Oracle package

```
!pip install cx_Oracle
```

Importing the libraries and setting the Oracle Client address

```
import cx_Oracle

import os

directory_path = os.getcwd() + "\instantclient-basic-windows.x64-21.3.0.0.0\instantclient_21_3"

cx_Oracle.init_oracle_client(lib_dir= directory_path)
```

Creating a connection to Oracle

```
my_username = "by1855j"

my_password = "by1855j"

connection = cx_Oracle.connect(user=my_username, password=my_password, dsn="OBIWAN")
```

Establishing the connection

```
from pandas import DataFrame

with connection.cursor() as cursor:

   cursor.execute("select WSENSOR_ID,LOCATION_ID,TIME_ID,RESULT from fact_table where
WSENSOR_ID=111 ")

   df = DataFrame(cursor.fetchall())

   df.columns = [x[0] for x in cursor.description]

   print("I got %d lines " % len(df))

df
```

Importing preprocessing from sklearn

```
from sklearn import preprocessing
```

Label Encoding

#converting each value in a column to a number

With this method, each value in the columns was converted to a number.

```
for column in df.columns:

      le = preprocessing.LabelEncoder()

      df[column] = le.fit_transform(df[column])
```

Splitting the data

The data should be separated into testing set and training set. The training set was used to build the KNN Classifier model and the test data was used to test the accuracy of the classifier. Splitting the data allows the KNN classification model to be optimised and overfitting to be avoided. As shown below, the test size is set to 0.3.

```
#independent variable array

X = df.iloc[:,1:3].values

#dependent variable vector

y = df.iloc[:,3].values
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=13)
```

Fitting KNN Algorithm model into the training set

In order to make a predictive model, the KNeighborsClassifier class was imported from the sklearn.neighbors library. This class was initialized by specifying the value of K with the parameter named n_neigbours. Since there is no fixed value for K, any value was given before trying different k values.

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=37)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)
```

Performing Cross-Validation

In order to obtain a higher accuracy score, the cross-validation method was applied. The small portion from the training dataset was named a validation dataset, and then different K values were evaluated to achieve proper accuracy.

To select the best parameters for KNN get the best-generalized data 23 fold cross-validation was performed as the following code segment.

```
from sklearn.model_selection import cross_val_score

neighbors=[]

cv_scores=[]

#performing 23-fold cross validation with K=13 for KNN

for k in range(1,53,4):

    neighbors.append(k)

    knn1=KNeighborsClassifier(n_neighbors = k)

    scores = cross_val_score(

      knn1,X_train,y_train,cv=23,scoring='accuracy')

    cv_scores.append(scores.mean())
```

```
scores=cross_val_score(knn1,X,y,cv=23,scoring='accuracy')

print(scores)
```

Finding the optimal number of K neighbors

The following code was used to find the K value that provides the highest accuracy score.

K= 37 is the optimal K value with misclassification error.

```
import matplotlib.pyplot as plt

#misclassification error

mse=[1-x for x in cv_scores]

#determining the best k value

optimal_k=neighbors[mse.index(min(mse))]

print("The optimal number of K neighbours = %d " %optimal_k)

#plotting misclassification error

plt.figure(figsize=(10,6))

plt.plot(neighbors,mse)

plt.xlabel("Number of K neighbors")

plt.ylabel("misclassification error")

plt.show()
```

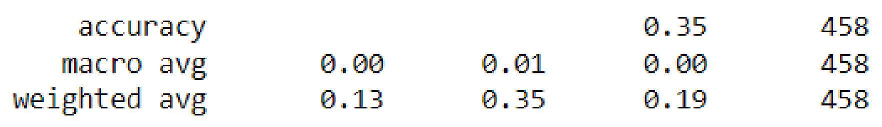
Evaluation

Classification Report

The classification method was used to measure the quality of the algorithm's predictions. It shows the metrics of the classification report.

```
from sklearn.metrics import classification_report

print("Classification Report : \n")

print(classification_report(y_test,y_pred))
```

output:



```
    accuracy                           0.35       458
   macro avg       0.00      0.01      0.00       458
weighted avg       0.13      0.35      0.19       458
```

Accuracy Score

By applying the KNN classifier model, the overall accuracy of the prediction was determined by the following code.

```
#accuracy

print(round(neigh.score(X_test, y_test),4)*100,'%')
```

Confusion Matrix

The confusion matrix was used as it gave a clear overview of the actual labels and the prediction of the model, but the desired result was not obtained due to the diversity of data in the class.

```
from sklearn.metrics import confusion_matrix

import seaborn as sns

#Summary of predictions

matrix=confusion_matrix(y_test,y_pred)

sns.heatmap(matrix.T,square=True,annot=True,fmt="d",cbar=False)

plt.title("Confusion Matrix")

plt.xlabel("test class")

plt.ylabel("predicted class")
```

Summary

The data to be classified with the KNN algorithm was tried to be classified according to the closeness relationship with the previous data. By applying and executing the KNN algorithm, it is aimed to make high-accuracy result estimations for the Ammonia(N) (111) water sensor based on location and time. The highest accuracy result obtained with this algorithm is 34.72%. In order to obtain more realistic results, the Linear Regression algorithm was also applied, but an unsatisfactory result was obtained such as r2_score= 7.139799480469533 %.

Different results for accuracy were obtained each time the number of neighbors was changed while applying the KNN classifier. Since there is no fixed value for the k variable, it was tested with different k values and an optimal k value was tried to be found. The model gave the highest accuracy when the optimal number of neighbors was 37, when the model was tested with the K value with values between 1 and 50. To select the best tuning parameter, 23-fold cross-validation was applied for the test where each fold contained 53 instances. The best result with this example was 0.34848485. This result shows that the classification made according to the result field does not reflect the efficiency to be obtained from the cross-validation well due to the inconsistency between the data.

Also, the classification report clearly represented the accuracy scores for the predicted model in detail. The confusion matrix was used as it gave a clear overview of the actual labels and the prediction of the model, but the desired result was not obtained due to the diversity of data in the class.

The most advantageous aspect of the KNN algorithm is that it is very simple to understand and easy to implement. Unlike the linear regression algorithm, it is a non-parametric algorithm, that is, there are assumptions that must be met to implement K-NN. When the algorithm receives new data, it compares the classes of the k nearest data to determine the class of the new data. However, Linear regression has many assumptions that must be met by the data before applying it. KNN is the most suitable option for this test as it is suitable for lower dimensional data. However, the study shows that despite its efficiency, the KNN algorithm may not give satisfactory results in large databases. As determining the number of neighbors to be used requires trial and error, it can negatively affect efficiency in large databases.