Data Analysis & Design

Database Implementation

Michael Cassar | 1HND6S

SECTION A | TABLE CREATION

P2.2 | P3.2

Script Start: Michael_Cassar_1HND6S_SectionA.SQL

Creating User Role

```
CREATE TABLE USERROLE
    (Role_id NUMBER(4)
        CONSTRAINT USERROLE_Role_id_PK
        PRIMARY KEY,
     Role_name VARCHAR2(25)
        CONSTRAINT USERROLE_Role_name_NN
        NOT NULL
        CONSTRAINT USERROLE_Role_name_UN
        UNIQUE);
```

Creating Country

```
CREATE TABLE COUNTRY
    (Country_id NUMBER(4)
        CONSTRAINT COUNTRY_Country_id_PK
        PRIMARY KEY,
     Country_name VARCHAR2(25)
        CONSTRAINT COUNTRY_Country_name_NN
        NOT NULL);
```

Creating Town

```
CREATE TABLE TOWN
    (Town_id NUMBER(4)
        CONSTRAINT TOWN_Town_id_PK
        PRIMARY KEY,
     CountryFK NUMBER(4)
        CONSTRAINT TOWN_CountryFK_FK
        REFERENCES COUNTRY(Country_id)
        ON DELETE CASCADE
        CONSTRAINT TOWN_CountryFK_NN
        NOT NULL);
```

Creating Item Category

```
CREATE TABLE ITEMCATEGORY
    (Category_id NUMBER(4)
        CONSTRAINT ITEMCATEGORY_Category_id_PK
        PRIMARY KEY,
     Category_name VARCHAR2(25)
        CONSTRAINT ITEMCATEGORY_Category_name_NN
        NOT NULL
        CONSTRAINT ITEMCATEGORY_Category_name_UN
        UNIQUE);
```

Creating Store

```
CREATE TABLE STORE
    (Store_id NUMBER(4)
        CONSTRAINT STORE_Store_id_PK
        PRIMARY KEY,
     Store_name VARCHAR2(25)
        CONSTRAINT STORE_Store_name_NN
        NOT NULL,
     TownFK NUMBER(4)
        CONSTRAINT STORE_TownFK_FK
        REFERENCES TOWN(Town_id)
        ON DELETE CASCADE
        CONSTRAINT STORE_TOWNFK_NN
        NOT NULL);
```

Creating Department

```
CREATE TABLE DEPARTMENT
    (Department_id NUMBER(4),
     Department_name VARCHAR2(25)
        CONSTRAINT DEPARTMENT_Department_name_NN
        NOT NULL,
     StoreFK NUMBER(4)
        CONSTRAINT DEPARTMENT_StoreFK_FK
        REFERENCES STORE(Store_id)
        ON DELETE CASCADE,
     CONSTRAINT DEPARTMENT_Composite_PK
     PRIMARY KEY (Department_id, StoreFK));
```

Creating Item

```
CREATE TABLE ITEM
    (Item_id NUMBER(4),
     Item_name VARCHAR2(25)
        CONSTRAINT ITEM_Item_name_NN
        NOT NULL,
     Item_price NUMBER(6,2)
        CONSTRAINT ITEM_Item_price_NN
        NOT NULL,
     Thumbnail BLOB,
     ItemCategoryFK NUMBER(4)
        CONSTRAINT ITEM_ItemCategoryFK_FK
        REFERENCES ITEMCATEGORY(Category_id)
        ON DELETE CASCADE
        CONSTRAINT ITEM_ItemCategoryFK_NN
        NOT NULL,
     DepartmentFK NUMBER(4),
     StoreFK NUMBER(4),
     CONSTRAINT ITEM_Composite_FK
     FOREIGN KEY(DepartmentFK, StoreFK)
     REFERENCES DEPARTMENT(Department_id, StoreFK)
     ON DELETE CASCADE,
     CONSTRAINT ITEM_Composite_PK
     PRIMARY KEY(Item_id, DepartmentFK, StoreFK));
```

```
CREATE TABLE USERACCOUNT
    (User_account_id NUMBER(4)
        CONSTRAINT USERACCOUNT_User_account_id_PK
        PRIMARY KEY,
     Username VARCHAR2(25)
        CONSTRAINT USERACCOUNT_Username_NN
        NOT NULL
        CONSTRAINT USERACCOUNT_Username_UN
        UNIQUE,
     Password VARCHAR2(25)
        CONSTRAINT USERACCOUNT_Password_NN
        NOT NULL,
     Secret_Question VARCHAR2(40)
        CONSTRAINT USERACCOUNT_Secret_Question_NN
        NOT NULL,
     SecretAnswer VARCHAR2(40)
        CONSTRAINT USERACCOUNT_SecretAnswer_NN
        NOT NULL,
     UserRoleFK NUMBER(4)
        CONSTRAINT USERACCOUNT_UserRoleFK_FK
        REFERENCES USERROLE(Role_id)
        ON DELETE CASCADE
        CONSTRAINT USERACCOUNT_UserRoleFK_NN
        NOT NULL);
```

Creating User Account

```
CREATE TABLE ORDERS
    (Order_id NUMBER(4)
        CONSTRAINT ORDERS_Order_id_PK
        PRIMARY KEY,
     Order_date DATE
        CONSTRAINT ORDERS_Order_date_NN
        NOT NULL,
     UserAccountFK NUMBER(4)
        CONSTRAINT ORDERS_UserAccountFK_FK
        REFERENCES USERACCOUNT(User_account_id)
        ON DELETE CASCADE
        CONSTRAINT ORDERS_UserAccountFK_NN
        NOT NULL);
```

Creating Orders

SCRIPT CONTINUES OVERLEAF

```
CREATE TABLE ORDERITEM
    (OrderFK NUMBER(4)
        CONSTRAINT ORDERITEM_OrderFK_FK
        REFERENCES ORDERS(Order_id)
        ON DELETE CASCADE,
     ItemFK NUMBER(4),
     DepartmentFK NUMBER(4),
     StoreFK NUMBER(4),
     Quantity NUMBER(6)
        CONSTRAINT ORDERITEM_Quantity_NN
        NOT NULL,
     CONSTRAINT ORDERITEM_Composite_FK
     FOREIGN KEY(ItemFK, DepartmentFK, StoreFK)
     REFERENCES ITEM(Item_id, DepartmentFK, StoreFK)
     ON DELETE CASCADE,
     CONSTRAINT ORDERITEM_Composite_PK
     PRIMARY KEY(OrderFK, ItemFK, DepartmentFK, StoreFK));
```

Creating Order Item

SCRIPT CONTINUES OVERLEAF

```
CREATE TABLE PERSON
    (Person_id NUMBER(4)
        CONSTRAINT PERSON_Person_id_PK
        PRIMARY KEY,
     Fullname VARCHAR2(40)
        CONSTRAINT PERSON_Fullname_NN
        NOT NULL,
     Residence_name VARCHAR2(25)
        CONSTRAINT PERSON_Residence_name_NN
        NOT NULL,
     Street VARCHAR2(40)
        CONSTRAINT PERSON_Street_NN
        NOT NULL,
     Town_name VARCHAR2(25)
        CONSTRAINT PERSON_Town_name_NN
        NOT NULL,
     Date_of_birth DATE
        CONSTRAINT PERSON_Date_of_birth_NN
        NOT NULL,
     Email VARCHAR2(25)
        CONSTRAINT PERSON_Email_NN
        NOT NULL
        CONSTRAINT PERSON_Email_UN
        UNIQUE,
     Contact_number VARCHAR2(25)
        CONSTRAINT PERSON_Contact_number_NN
        NOT NULL,
     PersonType VARCHAR2(25)
        CONSTRAINT PERSON_PersonType_NN
        NOT NULL,
     Registration_date DATE,
     Employment_date DATE,
     Vatnumber NUMBER(20)
        CONSTRAINT PERSON_Vatnumber_UN
        UNIQUE,
     UserAccountFK NUMBER(4)
        CONSTRAINT PERSON_UserAccountFK_FK
        REFERENCES USERACCOUNT(User_account_id)
        ON DELETE CASCADE
        CONSTRAINT PERSON_UserAccountFK_NN
        NOT NULL
        CONSTRAINT PERSON_UserAccountFK_UN
        UNIQUE,
     TownFK NUMBER(4)
        CONSTRAINT PERSON_TownFK_FK
        REFERENCES TOWN(Town_id)
        ON DELETE CASCADE
        CONSTRAINT PERSON_TownFK_NN
        NOT NULL);
```

Creating Person

Script End: Michael_Cassar_1HND6S_SectionA.SQL

SECTION B | NORMALISATION AND DESIGN IMPROVEMENTS

P1.3 | D1.1

PART 1

Script Start: Michael_Cassar_1HND6S_SectionB_Part1.SQL

Problem 1

```
ALTER TABLE PERSON
DROP COLUMN FULLNAME;

ALTER TABLE PERSON
ADD (FirstName VARCHAR2(25)
        CONSTRAINT PERSON_FirstName_NN
        NOT NULL,
     LastName VARCHAR2(25)
        CONSTRAINT PERSON_LastName_NN
        NOT NULL);
```

Problem 2

```
ALTER TABLE PERSON
DROP COLUMN TOWN_NAME;

ALTER TABLE TOWN
ADD (TownName VARCHAR2(25)
     CONSTRAINT TOWN_TownName_NN
     NOT NULL);
```

Problem 3

```
ALTER TABLE PERSON
DROP COLUMN PERSONTYPE;


ALTER TABLE PERSON
DROP COLUMN REGISTRATION_DATE;


ALTER TABLE PERSON
DROP COLUMN EMPLOYMENT_DATE;
```

SCRIPT CONTINUES OVERLEAF

```
CREATE TABLE CLIENT
    (ClientID NUMBER(4)
        CONSTRAINT CLIENT_ClientID_PK
        PRIMARY KEY,
     RegistrationDate DATE
        CONSTRAINT CLIENT_RegistrationDate_NN
        NOT NULL);


CREATE TABLE EMPLOYEE
    (EmployeeID NUMBER(4)
        CONSTRAINT EMPLOYEE_EmployeeID_PK
        PRIMARY KEY,
     EmploymentDate DATE
        CONSTRAINT EMPLOYEE_EmploymentDate_NN
        NOT NULL,
     ManagerID NUMBER(4)
        CONSTRAINT EMPLOYEE_ManagerID_FK
        REFERENCES PERSON(Person_id)
        ON DELETE CASCADE
        CONSTRAINT EMPLOYEE_ManagerID_UN
        UNIQUE);


ALTER TABLE PERSON
ADD (EmployeeFK NUMBER(4)
         CONSTRAINT PERSON_EmployeeFK_FK
         REFERENCES EMPLOYEE(EmployeeID)
         ON DELETE CASCADE
         CONSTRAINT PERSON_EmployeeFK_UN
         UNIQUE,
      ClientFK NUMBER(4)
         CONSTRAINT PERSON_ClientFK_FK
         REFERENCES CLIENT(ClientID)
         ON DELETE CASCADE
         CONSTRAINT PERSON_ClientFK_UN
         UNIQUE,
      CONSTRAINT PERSON_CheckSubType_CK
      CHECK ((EmployeeFK IS NOT NULL AND ClientFK IS NULL)
              OR
             (EmployeeFK IS NULL AND ClientFK IS NOT NULL)));
```

Problem 3

Script End: Michael_Cassar_1HND6S_SectionB_Part1.SQL

Analysis of Problem 3

There are two possible ways to implement the third problem. One way is by using subtypes, and the other is by using the exclusive relationship arc. The deciding factor to use subtypes in this case, is that the relationship arc should be used in the case where an employee or a client would not be a person. For example; A runway refers to an airport or a motor racing club however a runway is not an airport or a motor racing club. In this case an employee is a person and a client is also a person so subtypes should be used to solve the problem.

PART 2

Improvement Description

The ERD given depicts that, a Street is associated to each and every PERSON created in the database. Assuming that, over time, the database will get considerably larger, due to increase in CLIENTs and EMPLOYEEs, it might, and will most probably be apparent, that, the Street attribute will contain a substantial amount of anomalies, due to Street data repetitions. Not only that, but, if multiple STOREs exist in the same TOWN, the current database structure, will not allow the possibility of retrieving the Street these STOREs are actually based in.

In order to solve the above problems, another table will be created, STREET. This table will contain; StreetID, StreetName, Postcode, TownFK. In order to connect this entity appropriately, with PERSON and STORE, alterations will be made. The TownFKs will be removed from PERSON and STORE, and StreetFK will take their place. Postcode is included in the STREET table, as it may help the quality of the data saved, with regards to addresses, in certain situations. This attribute will be optional, in the case that the information is not provided to be inputted into the database. The Street attribute will then be removed from the PERSON table, as it would cause a further transitive dependency on the Primary Key.

SCRIPT SHOWN OVERLEAF

Script Start: Michael_Cassar_1HND6S_SectionB_Part2.SQL

```
ALTER TABLE PERSON
DROP COLUMN TownFK;


ALTER TABLE PERSON
DROP COLUMN STREET;


ALTER TABLE STORE
DROP COLUMN TownFK;
```

Dropping Foreign Keys and Street Column

```
CREATE TABLE STREET
    (StreetID NUMBER(4)
        CONSTRAINT STREET_StreetID_PK
        PRIMARY KEY,
     StreetName VARCHAR2(40)
        CONSTRAINT STREET_StreetName_NN
        NOT NULL,
     Postcode VARCHAR2(15),
     TownFK NUMBER(4)
        CONSTRAINT STREET_TownFK_FK
        REFERENCES TOWN(Town_id)
        ON DELETE CASCADE
        CONSTRAINT STREET_TownFK_NN
        NOT NULL);
```

Creating Street

```
ALTER TABLE PERSON
ADD (StreetFK NUMBER(4)
         CONSTRAINT PERSON_StreetFK_FK
         REFERENCES STREET(StreetID)
         ON DELETE CASCADE
         CONSTRAINT PERSON_StreetFK_NN
         NOT NULL);


ALTER TABLE STORE
ADD (StreetFK NUMBER(4)
         CONSTRAINT STORE_StreetFK_FK
         REFERENCES STREET(StreetID)
         ON DELETE CASCADE
         CONSTRAINT STORE_StreetFK_NN
         NOT NULL);
```

Adding Foreign Keys

Script End: Michael_Cassar_1HND6S_SectionB_Part2.SQL

SECTION C | ADDITIONAL SCHEMA OBJECTS

P3.1 | P4.5

Script Start: Michael_Cassar_1HND6S_SectionC.SQL

Creating Guest Accounts

```
CREATE FORCE VIEW GUESTACCOUNTS
AS
   SELECT username,
          password,
          role_id
   FROM USERACCOUNT UA
   JOIN USERROLE UR
   ON (UA.userrolefk = UR.role_id)
   WHERE LOWER(UR.role_name) = 'guest'
   WITH CHECK OPTION
      CONSTRAINT GUESTACCOUNTS_CO;
```

Creating Person Details

```
CREATE OR REPLACE FORCE VIEW PERSONDETAILS
AS
   SELECT p.firstname,
          p.lastname,
          p.date_of_birth,
          p.contact_number,
          p.residence_name,
          s.streetname,
          t.townname,
          co.country_name
   FROM PERSON p
   JOIN STREET s
   ON (p.streetfk = s.streetid)
   JOIN TOWN t
   ON (s.townfk = t.town_id)
   JOIN COUNTRY co
   ON (t.countryfk = co.country_id)
   WITH READ ONLY
      CONSTRAINT PERSONDETAILS_RO;
```

Script End: Michael_Cassar_1HND6S_SectionC.SQL

Benefits of User Views

Restricting Access

- ✓ Views can be used in order to restrict user access, by not allowing a specific user, with a specific user role, to access or modify data outside of that view. This provides a great level of security, as to the user, any other restricted view or object within the database, is non-existent, as the main portal for that user is only that view.

Data Independence

- ✓ Views can make use of sub-queries and joins. This means, that the query used to display data in the view, can retrieve data from multiple tables, allowing the possibility for the user, to have one single point of interaction with the database per view.

Present Different Views According to Their Purpose

- ✓ Since views are generally tied to a specific purpose, containing related data, it will be easier for users to make use of the data provided, since users may access that data, according to their needs.

SECTION D | TESTING

P4.1 | P4.4

Script Start: Michael_Cassar_1HND6S_SectionD.SQL

Creating Sequences

```
CREATE SEQUENCE USERROLE_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 10;


CREATE SEQUENCE USERACCOUNT_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 60;


CREATE SEQUENCE ORDER_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 60;


CREATE SEQUENCE ITEM_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 30;
```

SCRIPT CONTINUES OVERLEAF

```
CREATE SEQUENCE ITEMCATEGORY_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 10;


CREATE SEQUENCE DEPARTMENT_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 10;


CREATE SEQUENCE STORE_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 5;


CREATE SEQUENCE TOWN_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 10;


CREATE SEQUENCE COUNTRY_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 10;
```

Creating Sequences

SCRIPT CONTINUES OVERLEAF

```
CREATE SEQUENCE PERSON_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 60;


CREATE SEQUENCE EMPLOYEE_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 30;


CREATE SEQUENCE CLIENT_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 30;


CREATE SEQUENCE STREET_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 30;
```

Creating Sequences

```
INSERT INTO COUNTRY (country_id, country_name)
VALUES (COUNTRY_SEQUENCE.NEXTVAL, 'Malta');


INSERT INTO COUNTRY (country_id, country_name)
VALUES (COUNTRY_SEQUENCE.NEXTVAL, 'United Kingdom');


INSERT INTO COUNTRY (country_id, country_name)
VALUES (COUNTRY_SEQUENCE.NEXTVAL, 'Australia');
```

Country Inserts

SCRIPT CONTINUES OVERLEAF

```
INSERT INTO TOWN (town_id, countryfk, townname)
VALUES (TOWN_SEQUENCE.NEXTVAL, 1, 'Birzebbuga');


INSERT INTO TOWN (town_id, countryfk, townname)
 VALUES (TOWN_SEQUENCE.NEXTVAL, 2, 'Ealing');


INSERT INTO TOWN (town_id, countryfk, townname)
VALUES (TOWN_SEQUENCE.NEXTVAL, 3, 'Canberra');
```

Town Inserts

```
INSERT INTO STREET (streetid, streetname, postcode,
                    townfk)
VALUES (STREET_SEQUENCE.NEXTVAL, 'Wied il-Buni',
        'BBG2653', 1);


INSERT INTO STREET (streetid, streetname, postcode,
                    townfk)
VALUES (STREET_SEQUENCE.NEXTVAL, 'Fulmer Way',
        'W132DY', 2);


INSERT INTO STREET (streetid, streetname, postcode,
                    townfk)
 VALUES (STREET_SEQUENCE.NEXTVAL, 'Alinga Street',
        'ACT2601', 3);
```

Street Inserts

```
INSERT INTO USERROLE (role_id, role_name)
VALUES (USERROLE_SEQUENCE.NEXTVAL, 'Guest');


INSERT INTO USERROLE (role_id, role_name)
VALUES (USERROLE_SEQUENCE.NEXTVAL,
        'System Administrator');


INSERT INTO USERROLE (role_id, role_name)
 VALUES (USERROLE_SEQUENCE.NEXTVAL, 'Normal User');
```

User Role Inserts

SCRIPT CONTINUES OVERLEAF

```
INSERT INTO USERACCOUNT (user_account_id, username,
                         password, secret_question,
                         secretanswer, userrolefk)
VALUES (USERACCOUNT_SEQUENCE.NEXTVAL, 'JBorg',
       'Shh123', 'What is your mothers maiden name?',
       'Vella', 1);


INSERT INTO USERACCOUNT (user_account_id, username,
                         password, secret_question,
                         secretanswer, userrolefk)
VALUES (USERACCOUNT_SEQUENCE.NEXTVAL, 'FVella',
       'Quiet123', 'What is your fathers birth place?',
       'Pieta', 1);


INSERT INTO USERACCOUNT (user_account_id, username,
                         password, secret_question,
                         secretanswer, userrolefk)
VALUES (USERACCOUNT_SEQUENCE.NEXTVAL, 'MCassar',
        'Password123', 'Who was your first teacher?',
        'Ms. Bahram', 2);


INSERT INTO USERACCOUNT (user_account_id, username,
                         password, secret_question,
                         secretanswer, userrolefk)
VALUES (USERACCOUNT_SEQUENCE.NEXTVAL, 'RSaliba',
       'Secret123', 'What color was your first car?',
       'Yellow', 3);
```

User Account Inserts

```
INSERT INTO ITEMCATEGORY (category_id, category_name)
VALUES (ITEMCATEGORY_SEQUENCE.NEXTVAL, 'Electronics');


INSERT INTO ITEMCATEGORY (category_id, category_name)
VALUES (ITEMCATEGORY_SEQUENCE.NEXTVAL, 'Food');


INSERT INTO ITEMCATEGORY (category_id, category_name)
VALUES (ITEMCATEGORY_SEQUENCE.NEXTVAL, 'Clothing');
```

Item Category Inserts

SCRIPT CONTINUES OVERLEAF

```
INSERT INTO EMPLOYEE (employeeid, employmentdate,
                      managerid)
VALUES (EMPLOYEE_SEQUENCE.NEXTVAL, '18-JAN-07', NULL);


INSERT INTO PERSON (person_id, residence_name,
                    date_of_birth, email,
                    contact_number, vatnumber,
                    useraccountfk, firstname,
                    lastname, employeefk,
                    clientfk, streetfk)
VALUES (PERSON_SEQUENCE.NEXTVAL, 'Edera', '02-AUG-90',
        'michael.cassar@me.com', '+356 99579418',
        033103912208, 3, 'Michael', 'Cassar',
        EMPLOYEE_SEQUENCE.CURRVAL, NULL, 1);


INSERT INTO EMPLOYEE (employeeid, employmentdate,
                      managerid)
VALUES (EMPLOYEE_SEQUENCE.NEXTVAL, '23-JUN-09', 1);


INSERT INTO PERSON (person_id, residence_name,
                    date_of_birth, email,
                    contact_number, vatnumber,
                    useraccountfk, firstname,
                    lastname, employeefk,
                    clientfk, streetfk)
VALUES (PERSON_SEQUENCE.NEXTVAL, '12', '12-JAN-92',
        'rachel.saliba@live.com', '+44 77223457',
        109210910291, 4, 'Rachel', 'Saliba',
        EMPLOYEE_SEQUENCE.CURRVAL, NULL, 2);
```

Employee and Person Inserts

SCRIPT CONTINUES OVERLEAF

```
INSERT INTO CLIENT (clientid, registrationdate)
VALUES (CLIENT_SEQUENCE.NEXTVAL, '05-JUL-02');


INSERT INTO PERSON (person_id, residence_name,
                    date_of_birth, email,
                    contact_number, vatnumber,
                    useraccountfk, firstname,
                    lastname, employeefk,
                    clientfk, streetfk)
VALUES (PERSON_SEQUENCE.NEXTVAL, '136', '01-APR-91',
        'frans.vella@gmail.com', '+99 89221920',
        NULL, 2, 'Frans', 'Vella', NULL,
        CLIENT_SEQUE NCE.CURRVAL, 3);


INSERT INTO CLIENT (clientid, registrationdate)
VALUES (CLIENT_SEQUENCE.NEXTVAL, '01-MAY-06');


INSERT INTO PERSON (person_id, residence_name,
                    date_of_birth, email,
                    contact_number, vatnumber,
                    useraccountfk, firstname,
                    lastname, employeefk,
                    clientfk, streetfk)
VALUES (PERSON_SEQUENCE.NEXTVAL, 'Lapis Lazuli',
        '07-FEB-85', 'joeborg@yahoo.com',
        '+356 99443321', NULL, 1, 'Joe', 'Borg',
        NULL, CLIENT_SEQUENC E.CURRVAL, 1);
```

Client and Person Inserts

```
INSERT INTO STORE (store_id, store_name, streetfk)
VALUES (STORE_SEQUENCE.NEXTVAL, 'La Bonbonniere', 1);


INSERT INTO STORE (store_id, store_name, streetfk)
VALUES (STORE_SEQUENCE.NEXTVAL, 'DDS LTD', 2);


INSERT INTO STORE (store_id, store_name, streetfk)
VALUES (STORE_SEQUENCE.NEXTVAL, 'TechPoint Solutions', 3);
```

Store Inserts

SCRIPT CONTINUES OVERLEAF

```
INSERT INTO DEPARTMENT (department_id,
                        department_name,
                        storefk)
VALUES (DEPARTMENT_SEQUENCE.NEXTVAL, 'Sales', 1);


INSERT INTO DEPARTMENT (department_id,
                        department_name,
                        storefk)
VALUES (DEPARTMENT_SEQUENCE.NEXTVAL, 'e-Commerce', 2);


INSERT INTO DEPARTMENT (department_id,
                        department_name,
                        storefk)
VALUES (DEPARTMENT_SEQUENCE.NEXTVAL, 'Microchips', 3);
```

Department Inserts

```
INSERT INTO ITEM (item_id, item_name, item_price,
                  thumbnail, itemcategoryfk,
                  departmentfk, storefk)
VALUES (ITEM_SEQUENCE.NEXTVAL, 'MicrochipZXF',
        215.87, NULL, 1, 3, 3);


INSERT INTO ITEM (item_id, item_name, item_price,
                  thumbnail, itemcategoryfk,
                  departmentfk, storefk)
VALUES (ITEM_SEQUENCE.NEXTVAL, 'Pizza', 2.50,
        NULL, 2, 1, 1);


INSERT INTO ITEM (item_id, item_name, item_price,
                  thumbnail, itemcategoryfk,
                  departmentfk, storefk)
VALUES (ITEM_SEQUENCE.NEXTVAL,
        'Long Sleeve Shirt 73EF', 12.99, NULL,
        3, 2, 2);
```

Item Inserts

```
INSERT INTO ORDERS (order_id, order_date, useraccountfk)
VALUES (ORDER_SEQUENCE.NEXTVAL, '12-MAR-11', 2);


INSERT INTO ORDERS (order_id, order_date, useraccountfk)
VALUES (ORDER_SEQUENCE.NEXTVAL, '01-JAN-10', 1);


INSERT INTO ORDERS (order_id, order_date, useraccountfk)
VALUES (ORDER_SEQUENCE.NEXTVAL, '07-FEB-12', 3);
```

Orders Inserts

```
INSERT INTO ORDERITEM (orderfk, itemfk, departmentfk,
                       storefk, quantity)
VALUES (1, 1, 3, 3, 12);


INSERT INTO ORDERITEM (orderfk, itemfk, departmentfk,
                       storefk, quantity)
VALUES (2, 2, 1, 1, 1);


INSERT INTO ORDERITEM (orderfk, itemfk, departmentfk,
                       storefk, quantity)
VALUES (3, 3, 2, 2, 5);
```

Orderitem Inserts

```
COMMIT;
```

Committing Rows

```
INSERT INTO COUNTRY(country_id, country_name)
VALUES (COUNTRY_SEQUENCE.NEXTVAL, NULL);

INSERT INTO USERROLE(role_id, role_name)
VALUES (USERROLE_SEQUENCE.NEXTVAL, NULL);
```

Not Null Constraint

```
INSERT INTO CLIENT (clientid, registrationdate)
VALUES (CLIENT_SEQUENCE.NEXTVAL, '01-JUN-06');

INSERT INTO PERSON (person_id, residence_name,
                    date_of_birth, email,
                    contact_number, vatnumber,
                    useraccountfk, firstname,
                    lastname, employeefk, clientfk,
                    streetfk)
VALUES (PERSON_SEQUENCE.NEXTVAL, '704', '07-JUL-87',
        'joecassar@yahoo.com', '+356 99412121', NULL,
        1, 'Joe', 'Cassar', NULL,
        CLIENT_SEQUENCE.CURRVAL, 1);

INSERT INTO EMPLOYEE (employeeid, employmentdate,
                      managerid)
VALUES (EMPLOYEE_SEQUENCE.NEXTVAL, '23-JAN-09', 1);
```

Unique Constraint

SCRIPT CONTINUES OVERLEAF

```
INSERT INTO USERACCOUNT (user_account_id, username,
                         password, secret_question,
                         secretanswer, userrolefk)
VALUES (USERACCOUNT_SEQUENCE.NEXTVAL, 'RAxisa',
        'Silent123', 'What color was your first car?',
        'RED', 1);


INSERT INTO CLIENT (clientid, registrationdate)
VALUES (CLIENT_SEQUENCE.NEXTVAL, '01-FEB-07');


INSERT INTO EMPLOYEE (employeeid, employmentdate,
                      managerid)
VALUES (EMPLOYEE_SEQUENCE.NEXTVAL, '01-FEB-07', NULL);


INSERT INTO PERSON (person_id, residence_name,
                    date_of_birth, email,
                    contact_number, vatnumber,
                    useraccountfk, firstname,
                    lastname, employeefk,
                    clientfk, streetfk)
VALUES (PERSON_SEQUENCE.NEXTVAL, 'L-Arzella', '07-OCT-73',
        'ryanaxisa@yahoo.com', '+356 99443321', NULL,
        USERACCOUNT_SEQUENCE.CURRVAL, 'Ryan', 'Axisa',
        EMPLOYEE_SEQUENCE.CURRVAL, CLIENT_SEQUENCE.CURRVAL,
        1);


INSERT INTO USERACCOUNT (user_account_id, username,
                         password, secret_question,
                         secretanswer, userrolefk)
VALUES (USERACCOUNT_SEQUENCE.NEXTVAL, 'JCutajar',
        'Locked123', 'What color was your first car?',
        'Blue', 1);


INSERT INTO CLIENT (clientid, registrationdate)
VALUES (CLIENT_SEQUENCE.NEXTVAL, '01-JUL-02');


INSERT INTO EMPLOYEE (employeeid, employmentdate,
                      managerid)
VALUES (EMPLOYEE_SEQUENCE.NEXTVAL, '01-JUL-02', NULL);
```

Check Constraint

SCRIPT CONTINUES OVERLEAF

```
INSERT INTO PERSON (person_id, residence_name,
                    date_of_birth, email,
                    contact_number, vatnumber,
                    useraccountfk, firstname,
                    lastname, employeefk,
                    clientfk, streetfk)
VALUES (PERSON_SEQUENCE.NEXTVAL, '700', '07-SEP-71',
        'johncutajar@yahoo.com', '+356 99113112',
        NULL, USERACCOUNT_SEQUENCE.CURRVAL, 'John',
        'Cutajar', NULL, NULL, 1);
```

Check Constraint

```
INSERT INTO COUNTRY (country_id, country_name)
VALUES (NULL, 'Ireland');

INSERT INTO COUNTRY (country_id, country_name)
VALUES (1, 'Ireland');
```

Entity Integrity Constraint

```
ALTER TABLE TOWN
DROP CONSTRAINT TOWN_CountryFK_FK;

ALTER TABLE TOWN
MODIFY (CountryFK NUMBER(4)
           CONSTRAINT TOWN_COUNTRYFK_FK
           REFERENCES COUNTRY(Country_ID));

INSERT INTO TOWN (town_id, countryfk, townname)
VALUES (TOWN_SEQUENCE.NEXTVAL, 500, 'Missisipi');

DELETE FROM COUNTRY
WHERE Country_id = 1;

ALTER TABLE TOWN
DROP CONSTRAINT TOWN_CountryFK_FK;

ALTER TABLE TOWN
MODIFY (CountryFK NUMBER(4)
           CONSTRAINT TOWN_COUNTRYFK_FK
           REFERENCES COUNTRY(Country_ID)
           ON DELETE CASCADE);
```

Referential Integrity Constraint

Script End: Michael_Cassar_1HND6S_SectionD.SQL

Entity Integrity Constraint: Explained

Each tuple within a specific table must be represented by a Primary Key; that must implicitly be Unique, in order to identify each row individually. The Entity Integrity Constraint enforces a rule, which does not allow any Primary Key value to be null, or any Primary Key value to have the same value as another tuple, within the same table.

Referential Integrity Constraint: Explained

When a Parent table is referenced by a Child table, a relationship is created between the two tables. In order to maintain a level of consistency, any Foreign Key must refer to a tuple that is already existent in the parent table. A Parent table, or any other tuple referenced by a Child table, cannot be removed before the Child table releases this reference; unless handled by commands which nullify or delete the respective rows, upon deletion.

SECTION E | DATA DICTIONARY

P4.2 | P4.3

Person Insert Manual

In order to insert a person into the database, it's important to add the respective user account, and employee or client subtype/table values. The foreign keys of user account, and employee or client, must be added to the database beforehand; if this is not done, the person will not be inserted, and an oracle exception error will be thrown.

Consider the following six statements; the first collection of statements paves the insertion of an employee, and the second collection will insert a client. In order to explain how a person is inserted, it is a must for the directly referencing tables to be explained also.

```
INSERT INTO USERACCOUNT (user_account_id, username,
                         password, secret_question,
                         secretanswer, userrolefk)
VALUES (USERACCOUNT_SEQUENCE.NEXTVAL, 'CApap',
        'Shh123', 'What is your mothers maiden name?',
        'Cutajar', 3);


INSERT INTO EMPLOYEE (employeeid, employmentdate,
                      managerid)
VALUES (EMPLOYEE_SEQUENCE.NEXTVAL, '23-AUG-03', NULL);


INSERT INTO PERSON (person_id, residence_name,
                    date_of_birth, email,
                    contact_number, vatnumber,
                    useraccountfk, firstname,
                    lastname, employeefk, clientfk,
                    streetfk)
VALUES (PERSON_SEQUENCE.NEXTVAL, '113', '12-JAN-89',
        'christian.apap@live.com', '+44 77122457',
        103435910291, USERACCOUNT_SEQUENCE.CURRVAL,
        'Christian', 'Apap', EMPLOYEE_SEQUENCE.CURRVAL,
        NULL, 2);
```

First Insert

```
INSERT INTO USERACCOUNT (user_account_id, username,
                         password, secret_question,
                         secretanswer, userrolefk)
VALUES (USERACCOUNT_SEQUENCE.NEXTVAL, 'PPort',
        'Shh123', 'What is your favourite color?',
        'Blue', 1);


INSERT INTO CLIENT (clientid, registrationdate)
VALUES (CLIENT_SEQUENCE.NEXTVAL, '15-MAY-02');


INSERT INTO PERSON (person_id, residence_name,
                    date_of_birth, email,
                    contact_number, vatnumber,
                    useraccountfk, firstname,
                    lastname, employeefk,
                    clientfk, streetfk)
VALUES (PERSON_SEQUENCE.NEXTVAL, 'Serenity', '01-JUL-91',
        'peter.portelli@gmail.com', '+99 78290121', NULL,
        USERACCOUNT_SEQUENCE.CURRVAL, 'Peter',
        'Portelli', NULL, CLIENT_SEQUENCE.CURRVAL, 3);
```

Second Insert

User Account Inserts Explained

- ✓ The column names specified at the beginning of the statement, describe the data that is required to be inserted, in a particular tuple, corresponding to the user account table.

- ✓ The values required to match the specified attributes, may not be omitted, since they all have a not null constraint tied to them, this is why they must all be given a value.

- ✓ Username is a uniquely set column, so no two usernames can be the same.

- ✓ Consider the text that proceeds the values keyword, in the user account insert statement, each separated by a comma;

    - USERACCOUNT_SEQUENCE.NEXTVAL: This command specifies the usage of the next value in a sequence, created specifically for this table. The idea behind a sequence is, to always have a unique number inserted, to represent the tuple's primary key. This is done, in order to enforce the entity integrity constraint. Primary keys are generally handled by sequences; however, they can also be inputted manually. This attribute can only contain numbers that consist of a maximum of four digits. Data type: NUMBER(4).

- 'CApap': This input specifies the specific person's username. This input must be unique. This attribute can only contain characters up to a maximum length of 25. Data Type: VARCHAR2(25).

- 'Shh123': This input specifies the specific person's password. This attribute can only contain characters up to a maximum length of 25. Data Type: VARCHAR2(25).

- 'What is your mothers maiden name?': This input specifies the specific persons secret question. This attribute can only contain characters up to a maximum length of 40. Data Type: VARCHAR2(40).

- 'Cutajar': This input specifies the specific person's secret answer. This attribute can only contain characters up to a maximum length of 40. Data Type: VARCHAR2(40).

- 3: This input specifies the specific person's user role. This value is a foreign key, and must match a primary key value, specified in the user role table, in order to maintain referential integrity. The value inputted in our case, is 3, which refers to the value 'normal user', in the user role table. This attribute can only contain numbers that consist of a maximum of four digits, and that match a primary key in the user role table. Data Type: NUMBER(4).

Employee Inserts Explained

- ✓ The column names specified at the beginning of the statement, describe the data that is required to be inserted, in a particular tuple, corresponding to the employee table.

- ✓ The values required to match the specified attributes, may not be omitted, apart from the manager id attribute, since all attributes apart from manager id have a not null constraint tied to them. The manager id attribute, may be set to null, as this was specified in the specifications presented in the improvement stage of the database, due to the fact that certain employees may not have managers.

- ✓ The manager id attribute is also set to unique, since an employee may only be managed by one manager at a time, and a manager may only manage one employee at a time. This means, that there may be no foreign key repetitions in this attribute's column.

- ✓ Consider the text that proceeds the values keyword, in the employee insert statement, each separated by a comma;

    - EMPLOYEE_SEQUENCE.NEXTVAL: The idea behind sequences is explained above. The only difference in this case, is that we are making use of a sequence specific to the employee table. The value here can also be inputted manually. This attribute can only contain numbers that consist of a maximum of four digits. Data Type: NUMBER(4).

    - '23-AUG-03': This input specifies the specific person's employment date, which is an employee. This attribute can only contain dates. Data Type: DATE. Input Format: DD-MON-YY.

    - NULL: This input is set to null in this case as explained above, however, this attribute is specified as a foreign key which references the person table, and therefore, may hold the person id of a specific person which would be the manager of that particular employee. So if the input '1' was added instead of null, the person with person id '1' would be the manager of the employee, instead of having no manager. This attribute can only contain numbers that consist of a maximum of four digits. Data Type: NUMBER(4).

Client Inserts Explained

- ✓ The column names specified at the beginning of the statement, describe the data that is required to be inserted, in a particular tuple, corresponding to the client table.

- ✓ The values required to match the specified attributes, may not be omitted, since they all have a not null constraint tied to them, this is why they must all be given a value.

- ✓ Consider the text that proceeds the values keyword, in the client insert statement, each separated by a comma;

    - CLIENT_SEQUENCE.NEXTVAL: The idea behind sequences is explained above. The only difference in this case is that we are making use of a sequence specific to the client table. The value here can also be inputted manually. This attribute can only contain numbers that consist of a maximum of four digits. Data Type: NUMBER(4).

- '15-MAY-02': This input specifies the specific person's registration date, which is a client. This attribute can only contain dates. Data Type: DATE. Input Format: DD-MON-YY.

Person Inserts Explained

- ✓ The column names specified at the beginning of the statement, describe the data that is required to be inserted, in a particular tuple, corresponding to the person table.

- ✓ The values required to match the specified attributes, may not be omitted, apart from, employee fk or client fk and vat number, since all columns apart from the ones specified have a not null constraint tied to them.

- ✓ This table contains a check constraint, which only allows the usage of employee fk or client fk individually, meaning that one of these columns has to be null, in order to designate the specific person. Not only that, but the check constraint also enforces that one of the two attributes' data must be inputted, and both cannot be null. So for example; if employee fk has a value, then client must be null, meaning that the person is an employee, and vice-versa.

- ✓ In addition to the check constraint specified, both employee fk and client fk are set to unique, due to the fact that only one employee or only one client may be referred to by the person in question.

- ✓ Both Email and Vat number are uniquely set columns, so there cannot be two or more Emails and or two or more Vatnumbers having the same value.

- ✓ Consider the text that proceeds the values keyword, in the person insert statement, each separated by a comma;

    - PERSON_SEQUENCE.NEXTVAL: The idea behind sequences is explained above. The only difference in this case is that we are making use of a sequence specific to the person table. The value here can also be inputted manually. This attribute can only contain numbers that consist of a maximum of four digits. Data Type: NUMBER(4).

    - '113': This input specifies the specific person's residence name. This attribute can only contain characters up to a maximum length of 25. Data Type: VARCHAR2(25).

- '12-JAN-89': This input specifies the specific person's date of birth. This attribute can only contain dates. Data Type: DATE. Input Format: DD-MON-YY.

- 'christian.apap@live.com': This input specifies the specific person's email. This input must be unique. This attribute can only contain characters up to a maximum length of 25. Data Type: VARCHAR2(25).

- '+44 77122457': This input specifies the specific person's contact number. This attribute can only contain characters up to a maximum length of 25. Data Type: VARCHAR2(25).

- 103435910291: This input specifies the specific person's vat number. This attribute can only contain numbers up to a maximum length of 20. This value must be unique. This value may also be set to NULL, if the designated information is not provided. Data Type: NUMBER(20).

- USERACCOUNT_SEQUENCE.CURRVAL: This input specifies the specific person's user account. Since the useraccount sequence has been used directly before the execution of this insert statement, we can use .CURRVAL to retrieve the current value in that sequence, which will in-turn directly link the person to the appropriate account, through this foreign key attribute. This attribute can only contain numbers that consist of a maximum of four digits, and that match a primary key in the useraccount table. Data Type: NUMBER(4).

- 'Christian': This input specifies the specific person's first name. This attribute can only contain characters up to a maximum length of 25. Data Type: VARCHAR2(25).

- 'Apap': This input specifies the specific person's last name. This attribute can only contain characters up to a maximum length of 25. Data Type: VARCHAR2(25).

- EMPLOYEE_SEQUENCE.CURRVAL: This input designates the specific person as an employee. Since the employee sequence has been used directly before the execution of this insert statement, we can use .CURRVAL to retrieve the current value in that sequence, which will in-turn directly link the person to the appropriate record in the employee table, through this foreign key attribute. This attribute can only contain numbers that consist of a maximum of four digits,

and that match a primary key in the employee table. Data Type: NUMBER(4).

- NULL: As specified above, due to the check constraint, only one of employee fk or client fk can be inputted at a time. If the person were meant to be a client, employee fk would be set to null, and this attribute would be filled as mentioned above, however, using the client sequence instead. This attribute can only contain numbers that consist of a maximum of four digits, and that match a primary key in the client table. Data Type: NUMBER(4).

- 2: This input specifies the specific person's street. This value is a foreign key, and must match a primary key value, specified in the street table, in order to maintain referential integrity. The value inputted in our case, is 2, which refers to the values 'Fulmer Way' and 'W13 2DY', in the street table. This attribute can only contain numbers that consist of a maximum of four digits, and that match a primary key in the street table. Data Type: NUMBER(4).

Person Update Manual

Using the update statement, allows the manipulation, or changing of data, that has already been inserted into the particular table in the database. This may need to occur for a number of reasons, consider the following two statements;

```
UPDATE PERSON
SET residence_name = '112', vatnumber = 1129229990
WHERE person_id = 2;


UPDATE PERSON
SET residence_name = '112', date_of_birth = '12-JAN-83',
    email = 'michael.c@me.com',
    contact_number = '21659967'
WHERE person_id = 1;
```

Update Statements

In order to update the appropriate table it must be specified after UPDATE keyword. The first statement is updating the residence name, and the vat number of person id 2, and the second statement is updating the residence name, date of birth, email and contact number of person id 1. The SET keyword is used to include all the columns wanting to be modified to their specific values, however, matching the data types and constraints set at database creation. Columns can be added or omitted as seen fit, however, in order to update one particular person, the WHERE clause, ideally, must match a unique key, the primary key, else, if for

example, we compare to persons last name in the WHERE clause, we will update all the people whom have that last name. The WHERE clause may also contain sub-queries.

Person Delete Manual

Using the delete statement allows the deletion of tuples that have already been inserted into the particular table in the database. This may need to occur for a number of reasons, consider the following two statements;

```
DELETE FROM PERSON
WHERE last_name = 'Portelli';


DELETE FROM PERSON
WHERE person_id = 1;
```

Delete Statements

In order to delete from the appropriate table, it must be specified, after the DELETE FROM keywords. The first statement, is deleting every single row that contains a person with a last name, 'Portelli', and the second statement, is deleting the person whose person id, is 5. If a particular person is included as a manager, in another person's manager id column, in the respective employee table, they will not be deleted, due to the fact that their specific id is being used in that column. Since we are deleting the entirety of rows in this case, there is no need to specify any columns. The deciding factor on what to delete, is the WHERE clause, which can be used to specify which row shall be removed from the database. The WHERE clause may also contain sub-queries. Since employee and client are tied to person, depending on the person's sub type, they must also ideally be deleted using the foreign key present in the row wanting to be removed from the database, however this depends heavily on what is required to be deleted from the database.

Schema Object Documentation

Tables

Table Name: CLIENT		
Purpose: Created to implement the CLIENT subtype.		
	Attributes	
Name	**Data Type**	**Constraints**
CLIENTID	NUMBER(4)	PRIMARY KEY
REGISTRATIONDATE	DATE	NOT NULL

Table Name: COUNTRY		
Purpose: Created to store country details.		
Attributes		
Name	**Data Type**	**Constraints**
COUNTRY_ID	NUMBER(4)	PRIMARY KEY
COUNTRY_NAME	VARCHAR2(25)	NOT NULL

Table Name: DEPARTMENT		
Purpose: Created to store department details.		
Attributes		
Name	**Data Type**	**Constraints**
DEPARTMENT_ID	NUMBER(4)	PRIMARY KEY
DEPARTMENT_NAME	VARCHAR2(25)	NOT NULL
STOREFK	NUMBER(4)	PRIMARY KEY FOREIGN KEY

Table Name: EMPLOYEE		
Purpose: Created to implement the EMPLOYEE subtype.		
Attributes		
Name	**Data Type**	**Constraints**
EMPLOYEEID	NUMBER(4)	PRIMARY KEY
EMPLOYMENTDATE	DATE	NOT NULL
MANAGERID	NUMBER(4)	FOREIGN KEY UNIQUE

Table Name: ITEM		
Purpose: Created to store item details.		
Attributes		
Name	**Data Type**	**Constraints**
ITEM_ID	NUMBER(4)	PRIMARY KEY
ITEM_NAME	VARCHAR2(25)	NOT NULL
ITEM_PRICE	NUMBER(6,2)	NOT NULL
THUMBNAIL	BLOB	-----
ITEMCATEGORYFK	NUMBER(4)	FOREIGN KEY NOT NULL
DEPARTMENTFK	NUMBER(4)	PRIMARY KEY FOREIGN KEY
STOREFK	NUMBER(4)	PRIMARY KEY FOREIGN KEY

Table Name: ITEMCATEGORY		
Purpose: Created to store item category details.		
Attributes		
Name	**Data Type**	**Constraints**
CATEGORY_ID	NUMBER(4)	PRIMARY KEY
CATEGORY_NAME	VARCHAR2(25)	NOT NULL UNIQUE

Table Name: ORDERITEM		
Purpose: Created to implement a many to many relationship storing order item details.		
Attributes		
Name	**Data Type**	**Constraints**
ORDERFK	NUMBER(4)	PRIMARY KEY FOREIGN KEY
ITEMFK	NUMBER(4)	PRIMARY KEY FOREIGN KEY
DEPARTMENTFK	NUMBER(4)	PRIMARY KEY FOREIGN KEY
STOREFK	NUMBER(4)	PRIMARY KEY FOREIGN KEY
QUANTITY	NUMBER(6)	NOT NULL

Table Name: ORDERS		
Purpose: Created to store order details.		
Attributes		
Name	**Data Type**	**Constraints**
ORDER_ID	NUMBER(4)	PRIMARY KEY
ORDER_DATE	DATE	NOT NULL
USERACCOUNTFK	NUMBER(4)	FOREIGN KEY NOT NULL

Table Name: PERSON		
Purpose: Created to store person details.		
Attributes		
Name	**Data Type**	**Constraints**
PERSON_ID	NUMBER(4)	PRIMARY KEY
RESIDENCE_NAME	VARCHAR2(25)	NOT NULL
DATE_OF_BIRTH	DATE	NOT NULL
EMAIL	VARCHAR2(25)	NOT NULL UNIQUE
CONTACT_NUMBER	VARCHAR2(25)	NOT NULL
VATNUMBER	NUMBER(20)	UNIQUE
USERACCOUNTFK	NUMBER(4)	FOREIGN KEY NOT NULL UNIQUE
FIRSTNAME	VARCHAR2(25)	NOT NULL
LASTNAME	VARCHAR2(25)	NOT NULL
EMPLOYEEFK	NUMBER(4)	CHECK FOREIGN KEY UNIQUE
CLIENTFK	NUMBER(4)	CHECK FOREIGN KEY UNIQUE
STREETFK	NUMBER(4)	FOREIGN KEY NOT NULL

Table Name: STORE		
Purpose: Created to store store details.		
Attributes		
Name	**Data Type**	**Constraints**
STORE_ID	NUMBER(4)	PRIMARY KEY
STORE_NAME	VARCHAR2(25)	NOT NULL
STREETFK	NUMBER(4)	FOREIGN KEY NOT NULL

Table Name: STREET		
Purpose: Created to store street details.		
Attributes		
Name	**Data Type**	**Constraints**
STREETID	NUMBER(4)	PRIMARY KEY
STREETNAME	VARCHAR2(40)	NOT NULL
POSTCODE	VARCHAR2(15)	-----
TOWNFK	NUMBER(4)	FOREIGN KEY NOT NULL

Table Name: TOWN		
Purpose: Created to store town details.		
Attributes		
Name	**Data Type**	**Constraints**
TOWN_ID	NUMBER(4)	PRIMARY KEY
COUNTRYFK	NUMBER(4)	FOREIGN KEY NOT NULL
TOWNNAME	VARCHAR2(25)	NOT NULL

Table Name: USERACCOUNT		
Purpose: Created to store user account details.		
Attributes		
Name	**Data Type**	**Constraints**
USER_ACCOUNT_ID	NUMBER(4)	PRIMARY KEY
USERNAME	VARCHAR2(25)	NOT NULL UNIQUE
PASSWORD	VARCHAR2(25)	NOT NULL
SECRET_QUESTION	VARCHAR2(40)	NOT NULL
SECRETANSWER	VARCHAR2(40)	NOT NULL
USERROLEFK	NUMBER(4)	FOREIGN KEY NOT NULL

Table Name: USERROLE		
Purpose: Created to store user role details.		
Attributes		
Name	**Data Type**	**Constraints**
ROLE_ID	NUMBER(4)	PRIMARY KEY
ROLE_NAME	NUMBER(4)	NOT NULL UNIQUE

Sequences

Sequence Name: CLIENT_SEQUENCE
Purpose: Created to initiate a sequence for use with the CLIENT table's primary key.
First Value: 1
Increment: 1
Maximum Value: NOMINVALUE (1)
Minimum Value: NOMAXVALUE (999999999999999999999999999)
Cycle: NOCYCLE
Cache Amount: 30

Sequence Name: COUNTRY_SEQUENCE
Purpose: Created to initiate a sequence for use with the COUNTRY table's primary key.
First Value: 1
Increment: 1
Maximum Value: NOMINVALUE (1)
Minimum Value: NOMAXVALUE (999999999999999999999999999)
Cycle: NOCYCLE
Cache Amount: 10

Sequence Name: DEPARTMENT_SEQUENCE
Purpose: Created to initiate a sequence for use with the DEPARTMENT table's id.
First Value: 1
Increment: 1
Maximum Value: NOMINVALUE (1)
Minimum Value: NOMAXVALUE (999999999999999999999999999)
Cycle: NOCYCLE
Cache Amount: 10

Sequence Name: EMPLOYEE_SEQUENCE
Purpose: Created to initiate a sequence for use with the EMPLOYEE table's primary key.
First Value: 1
Increment: 1
Maximum Value: NOMINVALUE (1)
Minimum Value: NOMAXVALUE (999999999999999999999999999)
Cycle: NOCYCLE
Cache Amount: 30

Sequence Name: ITEM_SEQUENCE
Purpose: Created to initiate a sequence for use with the ITEM table's id.
First Value: 1
Increment: 1
Maximum Value: NOMINVALUE (1)
Minimum Value: NOMAXVALUE (999999999999999999999999999)
Cycle: NOCYCLE
Cache Amount: 30

Sequence Name: ITEMCATEGORY_SEQUENCE
Purpose: Created to initiate a sequence for use with the ITEMCATEGORY table's primary key.
First Value: 1
Increment: 1
Maximum Value: NOMINVALUE (1)
Minimum Value: NOMAXVALUE (999999999999999999999999999)
Cycle: NOCYCLE
Cache Amount: 10

Sequence Name: ORDER_SEQUENCE
Purpose: Created to initiate a sequence for use with the ORDER table's primary key.
First Value: 1
Increment: 1
Maximum Value: NOMINVALUE (1)
Minimum Value: NOMAXVALUE (999999999999999999999999999)
Cycle: NOCYCLE
Cache Amount: 60

Sequence Name: PERSON_SEQUENCE
Purpose: Created to initiate a sequence for use with the PERSON table's primary key.
First Value: 1
Increment: 1
Maximum Value: NOMINVALUE (1)
Minimum Value: NOMAXVALUE (999999999999999999999999999)
Cycle: NOCYCLE
Cache Amount: 60

Sequence Name: STORE_SEQUENCE
Purpose: Created to initiate a sequence for use with the STORE table's primary key.
First Value: 1
Increment: 1
Maximum Value: NOMINVALUE (1)
Minimum Value: NOMAXVALUE (999999999999999999999999999)
Cycle: NOCYCLE
Cache Amount: 5

Sequence Name: STREET_SEQUENCE
Purpose: Created to initiate a sequence for use with the STREET table's primary key.
First Value: 1
Increment: 1
Maximum Value: NOMINVALUE (1)
Minimum Value: NOMAXVALUE (999999999999999999999999999)
Cycle: NOCYCLE
Cache Amount: 30

Sequence Name: TOWN_SEQUENCE
Purpose: Created to initiate a sequence for use with the TOWN table's primary key.
First Value: 1
Increment: 1
Maximum Value: NOMINVALUE (1)

Minimum Value: NOMAXVALUE (9999999999999999999999999)
Cycle: NOCYCLE
Cache Amount: 10
Sequence Name: USERACCOUNT_SEQUENCE
Purpose: Created to initiate a sequence for use with the USERACCOUNT table's primary key.
First Value: 1
Increment: 1
Maximum Value: NOMINVALUE (1)
Minimum Value: NOMAXVALUE (9999999999999999999999999)
Cycle: NOCYCLE
Cache Amount: 60

Sequence Name: USERROLE_SEQUENCE
Purpose: Created to initiate a sequence for use with the USERROLE table's primary key.
First Value: 1
Increment: 1
Maximum Value: NOMINVALUE (1)
Minimum Value: NOMAXVALUE (9999999999999999999999999)
Cycle: NOCYCLE
Cache Amount: 10

Views

View Name: GUESTACCOUNTS
Purpose: Allows the viewing of accounts which have a guest role.
Description: This view puts together the username, password and role id of accounts which have a guest role.
Constraints: WITH CHECK OPTION | This view only allows inserts, updates and deletes, if the tuple as a member of the corresponding view is a Guest user.

View Name: PERSONDETAILS
Purpose: Allows the viewing of person details.
Description: This view puts together the name, surname, date of birth, contact number, residence, street, town and country of each and every person.
Constraints: WITH READ ONLY | This view only allows select statements to occur upon it.

SECTION F | PLANNING

D2.1

Script Start: Michael_Cassar_1HND6S_SectionF.SQL

```
ALTER TABLE TOWN
DROP CONSTRAINT TOWN_COUNTRYFK_FK;


ALTER TABLE TOWN
DROP CONSTRAINT TOWN_COUNTRYFK_NN;


ALTER TABLE TOWN
MODIFY (CountryFK NUMBER(4)
            CONSTRAINT TOWN_COUNTRYFK_FK
            REFERENCES COUNTRY(Country_ID)
            ON DELETE SET NULL);
```

On Delete Set Null

```
ALTER TABLE STREET
DROP CONSTRAINT STREET_TOWNFK_FK;


ALTER TABLE STREET
DROP CONSTRAINT STREET_TOWNFK_NN;


ALTER TABLE STREET
MODIFY (TownFK NUMBER(4)
            CONSTRAINT STREET_TOWNFK_FK
            REFERENCES TOWN(Town_ID)
            ON DELETE SET NULL);
```

On Delete Set Null

```
ALTER TABLE USERACCOUNT
DROP CONSTRAINT USERACCOUNT_USERROLEFK_FK;


ALTER TABLE USERACCOUNT
DROP CONSTRAINT USERACCOUNT_USERROLEFK_NN;


ALTER TABLE USERACCOUNT
MODIFY (UserRoleFK NUMBER(4)
            CONSTRAINT USERACCOUNT_USERROLEFK_FK
            REFERENCES USERROLE(Role_ID)
            ON DELETE SET NULL);
```

On Delete Set Null

Dropping Tables

```
DROP TABLE CLIENT CASCADE CONSTRAINTS;
DROP TABLE COUNTRY CASCADE CONSTRAINTS;
DROP TABLE DEPARTMENT CASCADE CONSTRAINTS;
DROP TABLE EMPLOYEE CASCADE CONSTRAINTS;
DROP TABLE ITEM CASCADE CONSTRAINTS;
DROP TABLE ITEMCATEGORY CASCADE CONSTRAINTS;
DROP TABLE ORDERITEM CASCADE CONSTRAINTS;
DROP TABLE ORDERS CASCADE CONSTRAINTS;
DROP TABLE PERSON CASCADE CONSTRAINTS;
DROP TABLE STORE CASCADE CONSTRAINTS;
DROP TABLE STREET CASCADE CONSTRAINTS;
DROP TABLE TOWN CASCADE CONSTRAINTS;
DROP TABLE USERACCOUNT CASCADE CONSTRAINTS;
DROP TABLE USERROLE CASCADE CONSTRAINTS;
```

Dropping Views

```
DROP VIEW GUESTACCOUNTS;
DROP VIEW PERSONDETAILS;
```

Dropping Sequences

```
DROP SEQUENCE CLIENT_SEQUENCE;
DROP SEQUENCE COUNTRY_SEQUENCE;
DROP SEQUENCE EMPLOYEE_SEQUENCE;
DROP SEQUENCE ITEM_SEQUENCE;
DROP SEQUENCE ITEMCATEGORY_SEQUENCE;
DROP SEQUENCE ORDER_SEQUENCE;
DROP SEQUENCE PERSON_SEQUENCE;
DROP SEQUENCE STORE_SEQUENCE;
DROP SEQUENCE STREET_SEQUENCE;
DROP SEQUENCE TOWN_SEQUENCE;
DROP SEQUENCE USERACCOUNT_SEQUENCE;
DROP SEQUENCE USERROLE_SEQUENCE;
DROP SEQUENCE DEPARTMENT_SEQUENCE;
```

Restoring Tables

```
FLASHBACK TABLE USERROLE TO BEFORE DROP;
FLASHBACK TABLE USERACCOUNT TO BEFORE DROP;
FLASHBACK TABLE TOWN TO BEFORE DROP;
FLASHBACK TABLE STREET TO BEFORE DROP;
FLASHBACK TABLE STORE TO BEFORE DROP;
FLASHBACK TABLE PERSON TO BEFORE DROP;
FLASHBACK TABLE ORDERS TO BEFORE DROP;
FLASHBACK TABLE ORDERITEM TO BEFORE DROP;
FLASHBACK TABLE ITEMCATEGORY TO BEFORE DROP;
FLASHBACK TABLE ITEM TO BEFORE DROP;
FLASHBACK TABLE EMPLOYEE TO BEFORE DROP;
FLASHBACK TABLE DEPARTMENT TO BEFORE DROP;
FLASHBACK TABLE COUNTRY TO BEFORE DROP;
FLASHBACK TABLE CLIENT TO BEFORE DROP;
```

```
CREATE FORCE VIEW GUESTACCOUNTS
AS
   SELECT username,
          password,
          role_id
   FROM USERACCOUNT UA
   JOIN USERROLE UR
   ON (UA.userrolefk = UR.role_id)
   WHERE LOWER(UR.role_name) = 'guest'
   WITH CHECK OPTION
      CONSTRAINT GUESTACCOUNTS_CO;


CREATE OR REPLACE FORCE VIEW PERSONDETAILS
AS
   SELECT p.firstname,
          p.lastname,
          p.date_of_birth,
          p.contact_number,
          p.residence_name,
          s.streetname,
          t.townname,
          co.country_name
   FROM PERSON p
   JOIN STREET s
   ON (p.streetfk = s.streetid)
   JOIN TOWN t
   ON (s.townfk = t.town_id)
   JOIN COUNTRY co
   ON (t.countryfk = co.country_id)
   WITH READ ONLY
      CONSTRAINT PERSONDETAILS_RO;
```

Re-Creating Views

```
CREATE SEQUENCE USERROLE_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 10;


CREATE SEQUENCE USERACCOUNT_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 60;
```

Re-Creating Sequences

```
CREATE SEQUENCE ORDER_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 60;


CREATE SEQUENCE ITEM_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 30;


CREATE SEQUENCE ITEMCATEGORY_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 10;


CREATE SEQUENCE DEPARTMENT_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 10;


CREATE SEQUENCE STORE_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 5;
```

Re-Creating Sequences

SCRIPT CONTINUES OVERLEAF

```
CREATE SEQUENCE TOWN_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 10;


CREATE SEQUENCE COUNTRY_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 10;


CREATE SEQUENCE PERSON_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 60;


CREATE SEQUENCE EMPLOYEE_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 30;


CREATE SEQUENCE CLIENT_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 30;
```

Re-Creating Sequences

SCRIPT CONTINUES OVERLEAF

```
CREATE SEQUENCE STREET_SEQUENCE
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   CACHE 30;
```

Re-Creating Sequences

Script End: Michael_Cassar_1HND6S_SectionF.SQL

SECTION G | PRESENTATION

M3.1

PLEASE FIND THE PRESENTATION ATTACHED OVERLEAF