GROUP 8

CSY2038 Assignment 2

Names:

J H

Stefan-Felician Baloiu – 20410488

L A

D F

EQUAL MARKS

Contents

[1. Plan 2](#_Toc93080731)

[1.1. Schema 2](#_Toc93080732)

[1.2. Skeleton Table 3](#_Toc93080733)

[1.3. Naming Conventions 4](#_Toc93080734)

[1.4. Queries 4](#_Toc93080735)

[1.5. Plan of Procedural Elements 5](#_Toc93080736)

[2. Proposed automation strategy 7](#_Toc93080737)

[3. Evidence of additional research 7](#_Toc93080738)

[4. Test Plan 7](#_Toc93080739)

[4.4. Scope 7](#_Toc93080740)

[4.5. Approach 7](#_Toc93080741)

[4.6. Resources 7](#_Toc93080742)

[4.7. Test items and features to be tested 7](#_Toc93080743)

[4.8. Schedule of test activities 7](#_Toc93080744)

[4.9. Contingency plans 7](#_Toc93080745)

[4.10. Test Case Planning 8](#_Toc93080746)

[5. References 11](#_Toc93080747)

[6. Appendix 11](#_Toc93080748)

[Appendix 1 - DROPS 11](#_Toc93080749)

[Appendix 2 - CREATES 12](#_Toc93080750)

[Appendix 3 - ALTERS 13](#_Toc93080751)

[Appendix 4 - INSERTS 14](#_Toc93080752)

[Appendix 5 – QUERIES 18](#_Toc93080753)

[Appendix 6 – FUNCTIONS and PROCEDURES 19](#_Toc93080754)

[Appendix 7 - PACKAGE 20](#_Toc93080755)

[Appendix 8 - TRIGGERS 21](#_Toc93080756)

[Appendix 9 – CURSOR 22](#_Toc93080757)

[APPENDIX 10 – Video Test statements 23](#_Toc93080758)

[Table 1 - Skeleton table 3](#_heading=h.3rdcrjn)

[Table 2 - Plan of procedural elements 5](#_heading=h.1ksv4uv)

[Table 3 - Test Table 7](#_heading=h.2bn6wsx)

[Figure 1 - Schema](about:blank) 2

# Plan

## Schema

Figure - Schema

retreat\_accommodation

retreat\_setting

programme

guest

accommodation

addresses

address\_type

review\_table\_type

review\_ type

practice\_type

ambassador\_type

ambassador\_varray\_type

*Figure SEQ Figure \\* ARABIC 1 - Schema*

## Skeleton Table

*Table 1 - Skeleton table*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tables** | **attribute** | **Key** | **Datatype** | **Constraints \**  **Defaults** |
| **addresses** | address\_type: | | | |
|  | country |  | VARCHAR2(25) |  |
|  | city |  | VARCHAR2(25) |  |
|  | street |  | VARCHAR2(25) |  |
|  | house\_number |  | VARCHAR2(5) |  |
|  | postcode |  | VARCHAR2(10) |  |
| **retreat\_settings** | retreat\_setting\_id | *pk* | NUMBER(6) | NOT NULL |
|  | retreat\_category |  | VARCHAR2(25) |  |
|  | setting |  | VARCHAR2(25) |  |
|  | reviews |  | reviews\_table\_type |  |
| **reviews\_table\_type** | reviews\_type: | | | |
|  | title |  | VARCHAR2(25) |  |
|
|  | rating |  | VARCHAR2(2) |  |
| **accommodations** | accommodation\_id | *pk* | NUMBER(6) | NOT NULL |
|  | accommodation\_style |  | VARCHAR2(25) |  |
|  | number\_of\_rooms |  | VARCHAR2(2) |  |
|  | price\_per\_night |  | NUMBER(6,2) |  |
|  | address |  | REF OF address\_type |  |
| **retreat\_accommodations** | retreat\_accommodation\_id | *pk* | NUMBER(6) | NOT NULL |
|  | retreat\_setting\_id | *FK* | NUMBER(6) | NOT NULL |
|  | accommodation\_id | *FK* | NUMBER(6) | NOT NULL |
| **programmes** | programme\_id | *pk* | NUMBER(6) | NOT NULL |
|  | cost |  | NUMBER(6,2) | ‘110.50’ |
|  | duration |  | VARCHAR2(15) | ‘1.5 HOURS’ |
|  | retreat\_accomodation\_id | *FK* | NUMBER(6) |  |
|  | guest\_id | FK | NUMBER(6) |  |
|  | ambassadors |  | ambassadors\_varray\_type |  |
| **practices\_type** | practice\_id |  | NUMBER(6) |  |
|  | practice\_name |  | VARCHAR2(25) |  |
| **ambassadors\_varray\_type** | ambassador\_type: | | | |
|  | ambassador\_id |  | NUMBER(6) |  |
|  | ambassador\_firstname |  | VARCHAR2(25) |  |
|  | ambassador\_surname |  | VARCHAR2(25) |  |
|  | date\_of\_birth |  | DATE |  |
|  | email |  | VARCHAR(50) |  |
|  | salary |  | NUMBER(8,2) |  |
|  | practice |  | practices\_type |  |
| **guests** | guest\_id | *pk* | NUMBER(6) | NOT NULL |
|  | guest\_firstname |  | VARCHAR2(25) | UPPER |
|  | guest\_surname |  | VARCHAR2(25) | UPPER |
|  | date\_of\_birth |  | DATE |  |
|  | phone\_number |  | VARCHAR2(25) | UNIQUE |
|  | email |  | VARCHAR2(50) | UNIQUE |
|  | ticket |  | CHAR |  |
|  | address |  | address\_type |  |

## Naming Conventions

* All entity names are singular
* All table names are plural
* All entities, tables and attributes do not include spaces
* All data and reserved words are upper case

## Queries

* Programme id of programmes costing under 200
* Guest id, name, email, phone number and address house number, postcode and country of guests from the UK
* Accommodation id with all address information for a specific accommodation
* Accommodation id and styles starting with ‘L’
* Outer join which shows guests, their accommodation and programmes they’re taking part in
* Inner join that shows just the accommodation IDs and retreat settings’ IDs used in the creation of retreat accommodations
* Ambassadors born after a certain date
* Count of every programme’s ambassadors salary that is over a certain value

## Plan of Procedural Elements

*Table 2 - Plan of procedural elements*

|  |  |  |
| --- | --- | --- |
|  | Name | Description |
| Function | func\_ambassador\_username | Function to create username for an ambassador. The function takes the first 2 characters of an ambassador’s first name and concatenates it with the first 5 characters of their surname. Both capitalized. |
| Function | func\_salary\_sum | Function to show total sum of salary paid to all ambassadors, rounded up. Then used in a package. |
| Function | func\_ticket\_ct | Function to count the number guests who have a ticket Y. Then used in a package. |
| Function | func\_rating\_avg | Function for average review rating for all retreat settings. Then used in a package. |
| Procedure | proc\_review\_deletion | Procedure to delete all reviews under a certain value. |
| Procedure | proc\_ambassador\_username | Procedure using func\_ambassador\_username, outputs a message “your login username is” and concatenates it with the username made in the username function. |
| Procedure | proc\_salary\_func | Procedure using func\_salary\_func. Outputs a message “The total sum of salary paid to ambassadors is” and concatenates it with the output of the salary sum function. Then used in a pacakge. |
| Procedure | proc\_ticket\_func | Procedure using func\_ticket\_ct. Outputs a message “There are” and “guests who have tickets” and concatenates it with the function that counts how many guests ‘Y’ have tickets. Then used in a package |
| Procedure | proc\_rating\_func | Procedure using func\_rating\_avg. Outputs a message “The average accommodation rating is” and concatenates its output of the rating function. Then used in a package. |
| Cursor | proc\_cur\_guests\_tickets | Cursor that sets the cost of a programme to 0 if the guest attending it has a ticket. Displays the name of the guest and concatenates it with an appropriate message. |
| Trigger | trig\_age\_category | Trigger that fires whenever a value is inserted or updated in the guest table which displays whether they’re an adult or a minor. |
| Trigger | trig\_invalid\_date | Trigger that fires whenever a date that’s higher than that of the system, an error message is displayed. |
| Package | funcs\_procs | Package that puts together all the small functions and procedures that query the database. |

# Proposed automation strategy

Functions, procedures, triggers, and cursors were implemented to automate the processing of data and any outputs.

# Evidence of additional research

Research was done on packaging to allow grouping of related PL/SQL subprograms. This was implemented into the script.

The following webpages were used:

* <https://www.oracletutorial.com/plsql-tutorial/plsql-package-specification/> 1
* <https://www.oracletutorial.com/plsql-tutorial/plsql-package-body/> 2
* <https://www.youtube.com/watch?v=lf5KsX8qaJs> 3

# Test Plan

## Scope

Show queries are producing the correct data and that procedural SQL is doing as it is expected. SQL does not need to be tested.

## Approach

Black box testing.

## Resources

Oracle connection

Laptop

Login and password

## Test items and features to be tested

* Query with substring – will allow creation of username in a procedure
* Query with deref table ref – to check table inserts
* Outer join query – to check table inserts
* Check procedural elements

## Schedule of test activities

* Testing query with substring from guest table
* Testing query with deref table ref
* Testing outer join query of accommodation id, retreat setting id or retreat accommodation id
* Testing procedure of review deletion, querying retreat setting and review nested table before and after

## Contingency plans

Created for any outcome not expected. Test case planning is in place to record steps taken to test the PL/SQL, therefore any amendments regarding testing noted. Backups of script file on Google drive, external hard drive and on personal computer.

## Test Case Planning

*Table 3 - Test Table*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Description | Expected Result | Actual Result | Action |
| 1 | SELECT  SUBSTR(guest\_firstname,1,2)|| SUBSTR(guest\_surname,1,5)  FROM guests  WHERE guest\_surname = 'BATES' ; | MABATES | ✔ |  |
| 2 | SELECT accommodation\_id, DEREF(address)  FROM accommodations  WHERE accommodation\_id = 202; | 202, 'UK', 'LONDON', 'GREENWICH ST', '10', 'EZ6N 7CD' | ✔ |  |
| 3 | SELECT ra.retreat\_accommodation\_id, a.accommodation\_id, rs.retreat\_setting\_id FROM retreat\_settings rs FULL OUTER JOIN retreat\_accommodations ra ON ra.retreat\_setting\_id = rs.retreat\_setting\_id FULL OUTER JOIN accommodations a ON ra.accommodation\_id = a.accommodation\_id; | Six rows returned, accommodation\_id 201 has no values in retreat\_setting\_id or retreat\_accommodation\_id | ✔ |  |
| 4 | SELECT MAX(reterat\_setting\_id) FROM retreat\_settings; | 104 | Error at line 1:  ORA-00904: “RETERAT\_SETTING\_ID”: invalid identifier | Correct the typo |
| 4a | SELECT MAX(retreat\_setting\_id) FROM retreat\_settings; | 104 | ✔ |  |
| 5 | SELECT retreat\_setting\_id, r.title, r.rating  FROM retreat\_settings rs, TABLE(rs.reviews) r; | 15 rows selected. | ✔ |  |
| 6 | DELETE TABLE(  SELECT rs.reviews  FROM retreat\_settings rs  WHERE retreat\_setting\_id = 100) r  WHERE r.rating < 6; | 0 rows deleted. | ✔ |  |
| 7 | SELECT retreat\_setting\_id, r.title, r.rating  FROM retreat\_settings rs, TABLE(rs.reviews) r; | 15 rows selected. | ✔ |  |
| 8 | DELETE TABLE(  SELECT rs.reviews  FROM retreat\_settings rs  WHERE retreat\_setting\_id = 101) r  WHERE r.rating < 6; | 1 row deleted. | ✔ |  |
| 9 | Execute procedure proc\_review\_deletion using values 100 for the retreat\_setting\_id and 6 for rating  exec proc\_review\_deletion(100, 6) | The reviews under the specified value have been deleted.  PL/SQL procedure successfully completed | ✔ |  |
| 10 | SELECT retreat\_setting\_id, r.title, r.rating  FROM retreat\_settings rs, TABLE(rs.reviews) r; | 11 rows selected. | 12 rows selected. | WHERE r.rating = in\_rating;  change to  WHERE r.rating < in\_rating; |
| 10a | SELECT retreat\_setting\_id, r.title, r.rating  FROM retreat\_settings rs, TABLE(rs.reviews) r; | 12 rows selected. | ✔ |  |
| 11 | Execute procedure proc\_review\_deletion using values 100 for the retreat\_setting\_id and 6 for rating  exec proc\_review\_deletion(100, 6) | The reviews under the specified value have been deleted.  PL/SQL procedure successfully completed | ✔ |  |
| 12 | SELECT retreat\_setting\_id, r.title, r.rating  FROM retreat\_settings rs, TABLE(rs.reviews) r; | 9 rows selected. | ✔ |  |
| 13 | Execute procedure using values 100 and 7  exec proc\_review\_deletion(100, 7) | There are no reviews with a rating lower than the selected value.  PL/SQL procedure successfully completed | ✔ |  |
| 14 | SELECT retreat\_setting\_id, r.title, r.rating  FROM retreat\_settings rs, TABLE(rs.reviews) r; | 9 rows selected. | ✔ |  |

# References

1 Oracle Tutorial. 2021. PL/SQL Package Specification. [ONLINE] Available at: <https://www.oracletutorial.com/plsql-tutorial/plsql-package-specification/>. [Accessed 10 January 2022].

2 Oracle Tutorial. 2021. PL/SQL Package Body. [ONLINE] Available at: <https://www.oracletutorial.com/plsql-tutorial/plsql-package-body/>. [Accessed 10 January 2022].

3 YouTube. 2018. Oracle - PL/SQL - Packages. [ONLINE] Available at: <https://www.youtube.com/watch?v=lf5KsX8qaJs>. [Accessed 10 January 2022].

# Appendix



/\*

CSY2038

Group 8

Joel Harsent - 20413335

Stefan-Felician Baloiu – 20410488

Louise Anderson – 20422051

Danielle Fox – 20422917

\*/

SET SERVEROUTPUT ON

## Appendix 1 - DROPS

-- DROPS --

DROP PROCEDURE proc\_review\_deletion;

DROP FUNCTION func\_ambassador\_username;

DROP PROCEDURE proc\_ambassador\_username;

DROP PACKAGE funcs\_procs;

DROP PROCEDURE proc\_cur\_guests\_tickets;

DROP TRIGGER trig\_age\_category;

DROP TRIGGER trig\_invalid\_date;

DROP TABLE programmes;

DROP TYPE ambassadors\_varray\_type;

DROP TYPE ambassadors\_type;

DROP TYPE practices\_type;

DROP TABLE guests;

DROP TABLE retreat\_accommodations;

DROP TABLE accommodations;

DROP TABLE addresses;

DROP TYPE address\_type;

DROP TABLE retreat\_settings;

DROP TYPE reviews\_table\_type;

DROP TYPE reviews\_type;

PURGE RECYCLEBIN;

## Appendix 2 - CREATES

-- CREATES --

CREATE TYPE reviews\_type AS OBJECT(

title VARCHAR2(25),

rating NUMBER(2) );

/

CREATE TYPE reviews\_table\_type AS TABLE OF reviews\_type;

/

CREATE TABLE retreat\_settings(

retreat\_setting\_id NUMBER(6) NOT NULL,

retreat\_category VARCHAR2(25),

setting VARCHAR2(25),

reviews reviews\_table\_type)

NESTED TABLE reviews STORE AS reviews\_table;

CREATE TYPE address\_type AS OBJECT(

country VARCHAR2(25),

city VARCHAR2(25),

street VARCHAR2(25),

house\_number VARCHAR2(5),

postcode VARCHAR2(10) );

/

CREATE TABLE addresses OF address\_type;

CREATE TABLE accommodations(

accommodation\_id NUMBER(6) NOT NULL,

accommodation\_style VARCHAR2(25),

number\_of\_rooms VARCHAR2(2),

price\_per\_night NUMBER(6,2),

address REF address\_type SCOPE IS addresses );

CREATE TABLE retreat\_accommodations(

retreat\_accommodation\_id NUMBER(6) NOT NULL,

retreat\_setting\_id NUMBER(6) NOT NULL,

accommodation\_id NUMBER(6) NOT NULL );

CREATE TABLE guests(

guest\_id NUMBER(6) NOT NULL,

guest\_firstname VARCHAR2(25),

guest\_surname VARCHAR2(25),

date\_of\_birth DATE,

phone\_number VARCHAR2(25),

email VARCHAR2(50),

ticket CHAR,

address address\_type );

CREATE TYPE practices\_type AS OBJECT(

practice\_id NUMBER(6),

practice\_name VARCHAR2(25) );

/

CREATE TYPE ambassadors\_type AS OBJECT(

ambassador\_id NUMBER(6),

ambassador\_firstname VARCHAR2(25),

ambassador\_surname VARCHAR2(25),

date\_of\_birth DATE,

email VARCHAR2(50),

salary NUMBER(8,2),

practice practices\_type );

/

CREATE TYPE ambassadors\_varray\_type AS VARRAY(40) OF ambassadors\_type;

/

CREATE TABLE programmes(

programme\_id NUMBER(6) NOT NULL,

cost NUMBER(6,2) DEFAULT '110.50',

duration VARCHAR2(15) DEFAULT '1.5 HOURS',

retreat\_accommodation\_id NUMBER(6),

guest\_id NUMBER(6),

ambassadors ambassadors\_varray\_type );

## Appendix 3 - ALTERS

-- ALTERS --

-- PKs

ALTER TABLE retreat\_settings

ADD CONSTRAINT pk\_retreat\_settings

PRIMARY KEY (retreat\_setting\_id);

ALTER TABLE accommodations

ADD CONSTRAINT pk\_accommodations

PRIMARY KEY (accommodation\_id);

ALTER TABLE retreat\_accommodations

ADD CONSTRAINT pk\_retreat\_accommodations

PRIMARY KEY (retreat\_accommodation\_id);

ALTER TABLE programmes

ADD CONSTRAINT pk\_programmes

PRIMARY KEY (programme\_id);

ALTER TABLE guests

ADD CONSTRAINT pk\_guests

PRIMARY KEY (guest\_id);

-- FKs

ALTER TABLE retreat\_accommodations

ADD CONSTRAINT fk\_ra\_retreat\_settings

FOREIGN KEY (retreat\_setting\_id)

REFERENCES retreat\_settings(retreat\_setting\_id);

ALTER TABLE retreat\_accommodations

ADD CONSTRAINT fk\_ra\_accommodations

FOREIGN KEY (accommodation\_id)

REFERENCES accommodations(accommodation\_id);

ALTER TABLE programmes

ADD CONSTRAINT fk\_p\_retreat\_accommodations

FOREIGN KEY (retreat\_accommodation\_id)

REFERENCES retreat\_accommodations(retreat\_accommodation\_id);

ALTER TABLE programmes

ADD CONSTRAINT fk\_p\_guests

FOREIGN KEY (guest\_id)

REFERENCES guests(guest\_id);

-- CHECKs

ALTER TABLE guests

ADD CONSTRAINT ck\_guest\_firstname

CHECK (guest\_firstname = upper(guest\_firstname));

ALTER TABLE guests

ADD CONSTRAINT ck\_guest\_surname

CHECK (guest\_surname = upper(guest\_surname));

-- UNIQUEs

ALTER TABLE guests

ADD CONSTRAINT uk\_email

UNIQUE (email);

ALTER TABLE guests

ADD CONSTRAINT uk\_phone\_number

UNIQUE (phone\_number);

## Appendix 4 - INSERTS

-- INSERTS --

INSERT INTO retreat\_settings(retreat\_setting\_id, retreat\_category, setting, reviews)

VALUES (100, 'REST AND RELAXATION', 'COASTAL',

reviews\_table\_type(

reviews\_type('INCREDIBLE', 9),

reviews\_type('SUPER RELAXING', 8),

reviews\_type('HAPPY', 8))

);

INSERT INTO retreat\_settings(retreat\_setting\_id, retreat\_category, setting, reviews)

VALUES (101, 'ENERGISING', 'PLANETARY LEY LINES',

reviews\_table\_type(

reviews\_type('AWESOME', 8),

reviews\_type('SPLENDID', 9),

reviews\_type('KINDA BAD', 5))

);

INSERT INTO retreat\_settings(retreat\_setting\_id, retreat\_category, setting, reviews)

VALUES (102, 'WEIGHT LOSS', 'WOODLAND',

reviews\_table\_type(

reviews\_type('THE VERY BEST', 10),

reviews\_type('MEDIOCRE', 6),

reviews\_type('BAD', 1))

);

INSERT INTO retreat\_settings(retreat\_setting\_id, retreat\_category, setting, reviews)

VALUES (103, 'ANXIETY RELIEF', 'COZY LODGE',

reviews\_table\_type(

reviews\_type('NEVER FELT SO RELAXED', 9),

reviews\_type('AMAZING ATMOSPHERE', 9),

reviews\_type('WAS OKAY', 5))

);

INSERT INTO retreat\_settings(retreat\_setting\_id, retreat\_category, setting, reviews)

VALUES (104, 'RELATIONSHIP COUNSELLING', 'TROPICAL',

reviews\_table\_type(

reviews\_type('DONE WONDERS', 10),

reviews\_type('BROKE UP', 6),

reviews\_type('AWFUL', 2))

);

INSERT INTO addresses(country, city, street, house\_number, postcode)

VALUES ('FRANCE', 'PARIS', 'JEANNE DARC ST', '55', '75000');

INSERT INTO addresses(country, city, street, house\_number, postcode)

VALUES ('FRANCE', 'PARIS', 'LOUIS XII ST', '43', '75008');

INSERT INTO addresses(country, city, street, house\_number, postcode)

VALUES ('UK', 'LONDON', 'GREENWICH ST', '10', 'EZ6N 7CD');

INSERT INTO addresses(country, city, street, house\_number, postcode)

VALUES ('GERMANY', 'HAMBURG', 'JAGER ST', '21', 'GR5 155');

INSERT INTO addresses(country, city, street, house\_number, postcode)

VALUES ('JAPAN', 'TOKYO', 'SENPAI ROAD', '33', 'TK3 51U');

INSERT INTO accommodations(accommodation\_id, accommodation\_style, number\_of\_rooms, price\_per\_night)

VALUES (200, 'CABIN', '4', 222.50);

UPDATE accommodations SET address =

(SELECT REF(a) FROM addresses a

WHERE a.street = 'JEANNE DARC ST')

WHERE accommodation\_id = 200;

INSERT INTO accommodations(accommodation\_id, accommodation\_style, number\_of\_rooms, price\_per\_night)

VALUES (201, 'LUXURY', '10', 999.99);

UPDATE accommodations SET address =

(SELECT REF(a) FROM addresses a

WHERE a.street = 'LOUIS XII ST')

WHERE accommodation\_id = 201;

INSERT INTO accommodations(accommodation\_id, accommodation\_style, number\_of\_rooms, price\_per\_night)

VALUES (202, 'LUXURY', '2', 400.00);

UPDATE accommodations SET address =

(SELECT REF(a) FROM addresses a

WHERE a.street = 'GREENWICH ST')

WHERE accommodation\_id = 202;

INSERT INTO accommodations(accommodation\_id, accommodation\_style, number\_of\_rooms, price\_per\_night)

VALUES (203, 'LODGE', '6', 360.00);

UPDATE accommodations SET address =

(SELECT REF(a) FROM addresses a

WHERE a.street = 'JAGER ST')

WHERE accommodation\_id = 203;

INSERT INTO accommodations(accommodation\_id, accommodation\_style, number\_of\_rooms, price\_per\_night)

VALUES (204, 'HUT', '5', 450.50);

UPDATE accommodations SET address =

(SELECT REF(a) FROM addresses a

WHERE a.street = 'SENPAI ROAD')

WHERE accommodation\_id = 204;

INSERT INTO retreat\_accommodations(retreat\_accommodation\_id, retreat\_setting\_id, accommodation\_id)

VALUES (300, 100, 202);

INSERT INTO retreat\_accommodations(retreat\_accommodation\_id, retreat\_setting\_id, accommodation\_id)

VALUES (301, 104, 202);

INSERT INTO retreat\_accommodations(retreat\_accommodation\_id, retreat\_setting\_id, accommodation\_id)

VALUES (302, 101, 200);

INSERT INTO retreat\_accommodations(retreat\_accommodation\_id, retreat\_setting\_id, accommodation\_id)

VALUES (303, 103, 203);

INSERT INTO retreat\_accommodations(retreat\_accommodation\_id, retreat\_setting\_id, accommodation\_id)

VALUES (304, 104, 204);

INSERT INTO guests(guest\_id, guest\_firstname, guest\_surname, date\_of\_birth, phone\_number, email, ticket, address)

VALUES (1000, 'JOHN', 'DOE', '25-DEC-1979', '+447712345834', 'JOHNDOE@EXAMPLE.COM', 'Y', address\_type('UK', 'LIVERPOOL', 'LIUER ST', '10', 'LV1 6LY'));

INSERT INTO guests(guest\_id, guest\_firstname, guest\_surname, date\_of\_birth, phone\_number, email, ticket, address)

VALUES (1001, 'MARY', 'SMITH', '15-MAR-1987', '+447756744567', 'MARYSMITH@EXAMPLE.COM', 'Y', address\_type('UK', 'GLASGOW', 'SHINY ST', '77', 'GG6 9HF'));

INSERT INTO guests(guest\_id, guest\_firstname, guest\_surname, date\_of\_birth, phone\_number, email, ticket, address)

VALUES (1002, 'NIA', 'VAUGHAN', '11-NOV-1991', '+447756355532', 'NIAVAUGHAN@EXAMPLE.COM', 'N', address\_type('UK', 'CARDIFF', 'SWANSEA ST', '2', 'CF22 5QT'));

INSERT INTO guests(guest\_id, guest\_firstname, guest\_surname, date\_of\_birth, phone\_number, email, ticket, address)

VALUES (1003, 'MARTIN', 'BATES', '06-OCT-2001', '+44777593764', 'MARTINBATES@EXAMPLE.COM', 'Y', address\_type('SINOH', 'PALLET CITY', 'GHASTLY ST', '3', 'GH1 311'));

INSERT INTO guests(guest\_id, guest\_firstname, guest\_surname, date\_of\_birth, phone\_number, email, ticket, address)

VALUES (1004, 'CHELSEA', 'JONES', '02-JUN-2005', '+447713068475', 'CHELSEAJONES@EXAMPLE.COM', 'N', address\_type('USA', 'WISCONSIN', 'BULL RD', '32', 'NY72 D11'));

INSERT INTO programmes(programme\_id, retreat\_accommodation\_id, guest\_id, ambassadors)

VALUES (400, 300, 1000,

ambassadors\_varray\_type(

ambassadors\_type(500, 'CHRIS', 'TYLER', '22-FEB-1994', 'CHRISTYLER@EXAMPLE.COM', 22400.50, practices\_type(600, 'HIIT')),

ambassadors\_type(501, 'ADAM', 'DAVIDSON', '07-DEC-1995', 'ADAMDAVIDSON@EXAMPLE.COM', 20986.50, practices\_type(601, 'MINDFULNESS')),

ambassadors\_type(502, 'SARAH', 'EVANS', '17-OCT-1998', 'SARAHEVANS@EXAMPLE.COM', 25400.00, practices\_type(602, 'CRYSTALS'))

)

);

INSERT INTO programmes(programme\_id, cost, duration, retreat\_accommodation\_id, guest\_id, ambassadors)

VALUES (401, 160.00, '2 HOURS', 301, 1002,

ambassadors\_varray\_type(

ambassadors\_type(503, 'JENNY', 'TYLER', '05-AUG-1989', 'JENNYTYLER@EXAMPLE.COM', 32500.00, practices\_type(603, 'YOGA')),

ambassadors\_type(504, 'LILY', 'WHITE', '11-SEP-1986', 'LILYWHITE@EXAMPLE.COM', 30000.00, practices\_type(604, 'TIBETAN SINGING BOWLS')),

ambassadors\_type(505, 'VIOLET', 'BLACK', '14-JUN-1980', 'VIOLETBLACK@EXAMPLE.COM', 29672.50, practices\_type(605, 'CLEANSING'))

)

);

INSERT INTO programmes(programme\_id, cost, duration, retreat\_accommodation\_id, guest\_id, ambassadors)

VALUES (402, 175.00, '2.5 HOURS', 300, 1000,

ambassadors\_varray\_type(

ambassadors\_type(506, 'DEBORAH', 'ADDISON', '20-JAN-1992', 'DEBORAHADDISSON@EXAMPLE.COM', 22500.00, practices\_type(600, 'HIIT')),

ambassadors\_type(507, 'KATSUROU', 'YAMASHITA', '14-APR-1990', 'KATSUROUYAMASHITA@EXAMPLE.COM', 36000.00, practices\_type(602, 'CRYSTALS')),

ambassadors\_type(508, 'OLGA-MARIE', 'ANIMUSPHERE', '06-MAY-1993', 'MARIEANIMUSPHERE@EXAMPLE.COM', 24500.50, practices\_type(603, 'YOGA'))

)

);

INSERT INTO programmes(programme\_id, cost, duration, retreat\_accommodation\_id, guest\_id, ambassadors)

VALUES (403, 200.00, '4 HOURS', 303, 1003,

ambassadors\_varray\_type(

ambassadors\_type(509, 'DANNI', 'JONES', '24-JAN-1973', 'DANNIJONES@EXAMPLE.COM', 28972.80, practices\_type(606, 'MEDIATION')),

ambassadors\_type(510, 'BEN', 'SHAW', '14-NOV-1999', 'BENSHAW@EXAMPLE.COM', 34270.10, practices\_type(607, 'BREATHING TECHNIQUES')),

ambassadors\_type(511, 'MARY', 'LINK', '23-FEB-1993', 'MARYLINK@EXAMPLE.COM', 29200.00, practices\_type(608, 'EXCERCISE'))

)

);

INSERT INTO programmes(programme\_id, cost, duration, retreat\_accommodation\_id, guest\_id, ambassadors)

VALUES (404, 500.00, '6 HOURS', 304, 1004,

ambassadors\_varray\_type(

ambassadors\_type(512, 'CAMMY', 'BURNS', '18-MAY-1990', 'CAMMYBURNS@EXAMPLE.COM', 52536.25, practices\_type(609, 'COMMUNICATION')),

ambassadors\_type(513, 'KATIE', 'LOWE', '25-APR-1983', 'KATIELOWE@EXAMPLE.COM', 38720.50, practices\_type(610, 'TRUST EXCERCISES')),

ambassadors\_type(514, 'KYLE', 'SANDERS', '02-OCT-2000', 'KYLESANDERS@EXAMPLE.COM', 39300.00, practices\_type(611, 'SOLVING ISSUES'))

)

);

## Appendix 5 – QUERIES

-- QUERIES --

-- Query to show programme id where cost is under 200

SELECT programme\_id "Programme ID"

FROM programmes

WHERE cost < 200;

-- Query to show guests contact information where guests live in the UK DESC

SELECT guest\_id "Guest ID", guest\_firstname "First name", guest\_surname "Surname", email "Email", phone\_number "Phone number", g.address.house\_number "House number", g.address.postcode "Postcode", g.address.country "Country"

FROM guests g

WHERE g.address.country = 'UK'

ORDER BY guest\_id DESC;

-- DEREF query to show accommodation id with address ref

SELECT accommodation\_id "Accommodation ID", DEREF(address) "Reference of Address"

FROM accommodations

WHERE accommodation\_id IN (

SELECT accommodation\_id

FROM accommodations

WHERE accommodation\_id = '203'

);

-- Query to show accommodation style and id of style strating with 'L'

SELECT accommodation\_id "Accommodation ID", accommodation\_style "Accommodation style"

FROM accommodations

WHERE accommodation\_style LIKE 'L%';

-- OUTER JOIN that shows guests and their accommodation and which programme they're taking part in

SELECT r.retreat\_accommodation\_id "Retreat accommodation ID", g.guest\_id "Guest ID", p.programme\_id "Programme ID", p.cost "Cost"

FROM retreat\_accommodations r

FULL OUTER JOIN programmes p

ON r.retreat\_accommodation\_id = p.retreat\_accommodation\_id

FULL OUTER JOIN guests g

ON g.guest\_id = p.guest\_id;

-- INNER JOIN that shows only the accommodation\_ids and retreat\_setting\_ids used in the creation of retreat\_accommodations

SELECT ra.retreat\_accommodation\_id "Retreat accommodation ID", a.accommodation\_id "Accommodation ID", rs.retreat\_setting\_id "Retreat setting ID"

FROM retreat\_settings rs

JOIN retreat\_accommodations ra

ON ra.retreat\_setting\_id = rs.retreat\_setting\_id

JOIN accommodations a

ON ra.accommodation\_id = a.accommodation\_id;

-- Query to show the ambassadors born after a certain date

SELECT programme\_id "Programme ID", a.ambassador\_id "Ambassador ID", a.ambassador\_firstname "First name", date\_of\_birth "DOB"

FROM programmes p, TABLE(p.ambassadors) a

WHERE date\_of\_birth > '01-JAN-1990'

ORDER BY a.ambassador\_id;

-- QUERY that displays the number of salaries of ambassadors over a threshold for every programme

SELECT programme\_id "Programme ID", COUNT(a.salary) "Number of salaries"

FROM programmes p, TABLE(p.ambassadors) a

WHERE a.salary > 30000

GROUP BY programme\_id;

## Appendix 6 – FUNCTIONS and PROCEDURES

-- FUNCTIONS and PROCEDURES --

-- Procedure to delete reviews under a certain value

CREATE OR REPLACE PROCEDURE proc\_review\_deletion(in\_retreat\_id retreat\_settings.retreat\_setting\_id%TYPE, in\_rating NUMBER) IS

vn\_retreat\_id NUMBER(6);

vn\_id\_value NUMBER(6);

BEGIN

SELECT MAX(retreat\_setting\_id)

INTO vn\_retreat\_id

FROM retreat\_settings;

vn\_id\_value := in\_retreat\_id;

WHILE vn\_id\_value <= vn\_retreat\_id LOOP

DELETE TABLE(

SELECT rs.reviews

FROM retreat\_settings rs

WHERE retreat\_setting\_id = vn\_id\_value) r

WHERE r.rating < in\_rating;

vn\_id\_value := vn\_id\_value +1;

END LOOP;

IF SQL%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('The reviews under the specified value have been deleted.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('There are no reviews with a rating lower than the selected value.');

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END proc\_review\_deletion;

/

-- Function to create a username for an ambassador

CREATE OR REPLACE FUNCTION func\_ambassador\_username(in\_ambassador\_id NUMBER) RETURN VARCHAR2 IS

vc\_username VARCHAR2(25);

vc\_firstname VARCHAR2(25);

vc\_surname VARCHAR2(25);

BEGIN

SELECT a.ambassador\_firstname, a.ambassador\_surname

INTO vc\_firstname, vc\_surname

FROM programmes p, TABLE(p.ambassadors) a

WHERE a.ambassador\_id = in\_ambassador\_id;

vc\_username := INITCAP(SUBSTR(vc\_firstname, 1,2)) || INITCAP(SUBSTR(vc\_surname, 1, 5));

RETURN vc\_username;

END func\_ambassador\_username;

/

-- Procedure using func\_ambassador\_username

CREATE OR REPLACE PROCEDURE proc\_ambassador\_username(in\_ambassador\_id NUMBER) IS

vc\_username VARCHAR(25);

BEGIN

vc\_username := func\_ambassador\_username(in\_ambassador\_id);

DBMS\_OUTPUT.PUT\_LINE('Your login username is ' || vc\_username || '.');

END proc\_ambassador\_username;

/

## Appendix 7 - PACKAGE

-- PACKAGE --

-- Creating a package with the titles of small functions and the procedures that call them

CREATE OR REPLACE PACKAGE funcs\_procs IS

FUNCTION func\_salary\_sum RETURN NUMBER; -- Creating a function that returns the sum of the salaries paid to ambassadors

PROCEDURE proc\_salary\_func; -- The procedure that calls the function and displays a message

FUNCTION func\_ticket\_ct RETURN NUMBER; -- Creating a function that returns the number of guests that have tickets

PROCEDURE proc\_ticket\_func; -- The procedure that calls the function and displays a message

FUNCTION func\_rating\_avg RETURN NUMBER; -- Creating a function that returns the average of a retreats ratings

PROCEDURE proc\_rating\_func; -- The procedure that calls the function and displays a message

END funcs\_procs;

/

-- The package body containing the functions and procedures

CREATE OR REPLACE PACKAGE BODY funcs\_procs IS

FUNCTION func\_salary\_sum RETURN NUMBER IS

vn\_salary\_sum NUMBER(9,2);

BEGIN

SELECT CEIL(SUM(a.salary))

INTO vn\_salary\_sum

FROM programmes p, TABLE(p.ambassadors) a;

RETURN vn\_salary\_sum;

END func\_salary\_sum;

PROCEDURE proc\_salary\_func IS

vn\_sum\_salary NUMBER(9,2);

BEGIN

vn\_sum\_salary := func\_salary\_sum;

DBMS\_OUTPUT.PUT\_LINE('The total sum of salary paid to ambassadors is ' || vn\_sum\_salary || '.');

END proc\_salary\_func;

FUNCTION func\_ticket\_ct RETURN NUMBER IS

vn\_counter\_ct NUMBER(3);

BEGIN

SELECT COUNT(guest\_id)

INTO vn\_counter\_ct

FROM guests

WHERE ticket = 'Y';

RETURN vn\_counter\_ct;

END func\_ticket\_ct;

PROCEDURE proc\_ticket\_func IS

vn\_no\_of\_tickets NUMBER(3);

BEGIN

vn\_no\_of\_tickets := func\_ticket\_ct;

DBMS\_OUTPUT.PUT\_LINE('There are ' || vn\_no\_of\_tickets || ' guests who have tickets.');

END proc\_ticket\_func;

FUNCTION func\_rating\_avg RETURN NUMBER IS

vn\_rating\_avg NUMBER(2);

BEGIN

SELECT AVG(r.rating)

INTO vn\_rating\_avg

FROM retreat\_settings rs, TABLE(rs.reviews) r;

RETURN vn\_rating\_avg;

END func\_rating\_avg;

PROCEDURE proc\_rating\_func IS

vn\_avg\_rating NUMBER(2);

BEGIN

vn\_avg\_rating := func\_rating\_avg;

DBMS\_OUTPUT.PUT\_LINE('The average accommodation rating is ' || vn\_avg\_rating);

END proc\_rating\_func;

END;

/

## Appendix 8 - TRIGGERS

-- TRIGGERS --

-- Trigger that displays whether a guest is an adult or a minor

CREATE OR REPLACE TRIGGER trig\_age\_category

AFTER INSERT OR UPDATE OF date\_of\_birth

ON guests

FOR EACH ROW

WHEN (NEW.date\_of\_birth IS NOT NULL)

DECLARE

vn\_age NUMBER(5,2);

BEGIN

vn\_age := MONTHS\_BETWEEN(SYSDATE, :NEW.date\_of\_birth)/12;

IF vn\_age < 18 THEN

DBMS\_OUTPUT.PUT\_LINE('The person is a minor aged ' || vn\_age || '.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('The person is an adult aged ' ||vn\_age || '.');

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END trig\_age\_category;

/

-- Trigger that displays an error message in case of an invalid birth date

CREATE OR REPLACE TRIGGER trig\_invalid\_date

AFTER INSERT OR UPDATE OF date\_of\_birth

ON guests

FOR EACH ROW

WHEN(NEW.date\_of\_birth>SYSDATE)

BEGIN

RAISE\_APPLICATION\_ERROR(-20000, 'Invalid date of birth');

END;

/

## Appendix 9 – CURSOR

-- CURSORS --

-- Cursor that modifies the cost of the programme a guest is attending to 0 if that customer has a ticket for it

CREATE OR REPLACE PROCEDURE proc\_cur\_guests\_tickets IS

CURSOR cur\_guests IS

SELECT guest\_id, guest\_firstname, guest\_surname, email, phone\_number, ticket

FROM guests;

BEGIN

FOR rec\_cur\_guests IN cur\_guests LOOP

IF rec\_cur\_guests.ticket = 'Y' THEN

UPDATE programmes SET cost = 0 WHERE guest\_id = rec\_cur\_guests.guest\_id;

IF SQL%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE(INITCAP(rec\_cur\_guests.guest\_firstname) || ' ' || INITCAP(rec\_cur\_guests.guest\_surname) || ' has a ticket so they don''t have to pay for their programme.');

ELSE

DBMS\_OUTPUT.PUT\_LINE(INITCAP(rec\_cur\_guests.guest\_firstname) || ' ' || INITCAP(rec\_cur\_guests.guest\_surname) || ' has a ticket but they''re not participating in any programmes.');

END IF;

END IF;

END LOOP;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END proc\_cur\_guests\_tickets;

/

## APPENDIX 10 – Video Test statements

-- Video commands to run --

/\*

COLUMN object\_name FORMAT A30

COLUMN object\_type FORMAT A12

SELECT TNAME FROM TAB;

SELECT object\_name, object\_type FROM user\_objects;

SELECT retreat\_setting\_id, r.title, r.rating

FROM retreat\_settings rs, TABLE(rs.reviews) r;

exec proc\_review\_deletion(100, 6)

SELECT retreat\_setting\_id, r.title, r.rating

FROM retreat\_settings rs, TABLE(rs.reviews) r;

SELECT a.ambassador\_id, a.ambassador\_firstname, a.ambassador\_surname

FROM programmes p, TABLE(p.ambassadors) a;

exec proc\_ambassador\_username(508)

SELECT a.ambassador\_id, a.salary

FROM programmes p, TABLE(p.ambassadors) a;

exec funcs\_procs.proc\_salary\_func

SELECT guest\_id, ticket

FROM guests;

exec funcs\_procs.proc\_ticket\_func

SELECT retreat\_setting\_id, r.rating

FROM retreat\_settings rs, TABLE(rs.reviews) r;

exec funcs\_procs.proc\_rating\_func

INSERT INTO guests(guest\_id, guest\_firstname, date\_of\_birth)

VALUES (1010, 'STEFAN', '06-OCT-2001');

INSERT INTO guests(guest\_id, guest\_firstname, date\_of\_birth)

VALUES (1011, 'IRRELEVANT', '10-JUL-2025');

SELECT g.guest\_id, g.guest\_firstname, g.guest\_surname, g.ticket, p.programme\_id, p.cost

FROM guests g

JOIN programmes p

ON g.guest\_id = p.guest\_id;

COLUMN guest\_firstname FORMAT A15

COLUMN guest\_surname FORMAT A15

exec proc\_cur\_guests\_tickets

DROP PROCEDURE proc\_review\_deletion;

DROP FUNCTION func\_ambassador\_username;

DROP PROCEDURE proc\_ambassador\_username;

DROP PACKAGE funcs\_procs;

DROP PROCEDURE proc\_cur\_guests\_tickets;

DROP TRIGGER trig\_age\_category;

DROP TRIGGER trig\_invalid\_date;

DROP TABLE programmes;

DROP TYPE ambassadors\_varray\_type;

DROP TYPE ambassadors\_type;

DROP TYPE practices\_type;

DROP TABLE guests;

DROP TABLE retreat\_accommodations;

DROP TABLE accommodations;

DROP TABLE addresses;

DROP TYPE address\_type;

DROP TABLE retreat\_settings;

DROP TYPE reviews\_table\_type;

DROP TYPE reviews\_type;

PURGE RECYCLEBIN;

SELECT TNAME FROM TAB;

SELECT object\_name, object\_type FROM user\_objects;

\*/