Southern Alberta Institute of Technology

**Database Creation for Travel Experts Agency**

by

**Femi Adams**

Supervised by

David Haas

Masha Amini

Winter 2023

Contents

[Workshop 1 9](#_Toc134141870)

[**Day 1 Interview questions** 9](#_Toc134141871)

[**Day 2 Interview questions** 10](#_Toc134141872)

[Workshop 2 12](#_Toc134141873)

[**Entity Relationship Diagram (ERD)** 12](#_Toc134141874)

[ERD documentation table 13](#_Toc134141875)

[Workshop 3 26](#_Toc134141876)

[**Disk configuration blueprint** 26](#_Toc134141877)

[Steps to create the database. 27](#_Toc134141878)

[Workshop 4 33](#_Toc134141879)

[**DBA User Creation** 33](#_Toc134141880)

[Tablespace creation 33](#_Toc134141881)

[Workshop 5 78](#_Toc134141882)

[**Update/progress of this project to TEA management** 78](#_Toc134141883)

[**User Creation** 78](#_Toc134141884)

[**User creation and granted roles** 89](#_Toc134141885)

[**Grant role to users** 91](#_Toc134141886)

[**Profile creation** 93](#_Toc134141887)

[Assign profile to users. 94](#_Toc134141888)

[Workshop 6 98](#_Toc134141889)

[**Configuring listener.ora file** 98](#_Toc134141890)

[**Configuring Tnsnames.ora File** 99](#_Toc134141891)

[**Configuring sqlnet.ora file** 100](#_Toc134141892)

[Workshop 7 102](#_Toc134141893)

[**Data Warehouse & Data Mining** 102](#_Toc134141894)

[Data Warehouse Model/Scripts for Summary Views: 102](#_Toc134141895)

[Metadata: 102](#_Toc134141896)

[Sample Reports: 102](#_Toc134141897)

[Marketing-Related Questions: 102](#_Toc134141898)

[Recommendations for Future Data Collection: 102](#_Toc134141899)

[Workshop 8 103](#_Toc134141900)

[**Travel agency backup plan** 103](#_Toc134141901)

[Frequency of backups: 103](#_Toc134141902)

[Recommended backup method: 103](#_Toc134141903)

[Recommendation concerning Archive/No archive mode with reasons: 104](#_Toc134141904)

[Recommendations concerning the number of redo log groups/members with reasons: 104](#_Toc134141905)

[Strategy for effective use of a tape drive (or separate device/location): 104](#_Toc134141906)

[Exact backup schedule: 104](#_Toc134141907)

[**Testing backup plan (Cold Backup & RMAN)** 104](#_Toc134141908)

[Cold backup 104](#_Toc134141909)

[RMAN Backup 106](#_Toc134141910)

[Workshop 9 107](#_Toc134141911)

[**Sample invoice/itinerary script** 107](#_Toc134141912)

[**Commission report script** 111](#_Toc134141913)

[**Entry of a new customer** 112](#_Toc134141914)

[Workshop – 10 114](#_Toc134141915)

[**Performance & Tuning** 114](#_Toc134141916)

[Workshop 11 115](#_Toc134141917)

[Cover letter 115](#_Toc134141918)

[Executive summary 116](#_Toc134141919)

[Background 116](#_Toc134141920)

[Business objectives 116](#_Toc134141921)

[Project objectives 117](#_Toc134141922)

[Scope 117](#_Toc134141923)

[Database System: 117](#_Toc134141924)

[Hardware Solution: 117](#_Toc134141925)

[Reasons for Selection: 118](#_Toc134141926)

[**Constraints** 118](#_Toc134141927)

[Budget: 118](#_Toc134141928)

[Timeframe: 118](#_Toc134141929)

[Resources: 118](#_Toc134141930)

[Assumptions 119](#_Toc134141931)

[Approach 119](#_Toc134141932)

[Requirements gathering: 119](#_Toc134141933)

[Development: 119](#_Toc134141934)

[Testing: 119](#_Toc134141935)

[Deployment: 119](#_Toc134141936)

[User training: 119](#_Toc134141937)

[Deliverables 120](#_Toc134141938)

[Requirements document: 120](#_Toc134141939)

[Design document: 120](#_Toc134141940)

[Database prototype: 120](#_Toc134141941)

[Testing report: 120](#_Toc134141942)

[User manual: 120](#_Toc134141943)

[User training: 120](#_Toc134141944)

[Project Schedule 120](#_Toc134141945)

[Budget 121](#_Toc134141946)

[Hardware expenses 121](#_Toc134141947)

[Software expenses: 121](#_Toc134141948)

[Personnel expenses: 121](#_Toc134141949)

[Other expenses: 122](#_Toc134141950)

[Recommendations 122](#_Toc134141951)

[Next Steps/ Progress Reporting 123](#_Toc134141952)

[Appendix 123](#_Toc134141953)

[Workshop 13 124](#_Toc134141954)

[**Oracle Hot Backup Shell Script and a datapump backup script** 124](#_Toc134141955)

[Datapump backup script (oradp.sh) 124](#_Toc134141956)

[Hot backup shell script (orahot.sh) 127](#_Toc134141957)

[REFERENCES 137](#_Toc134141958)

[**Workshop 3** 137](#_Toc134141959)

List of Figures

[**Figure1. 1 ERD of Travel Expert Agency 8**](#_Toc133790471)

[**Figure 2. 1 Init.ora file** 27](#_Toc134141960)

[**Figure 2. 2 Init.ora directory** 27](#_Toc134141961)

[**Figure 2. 3 Create TEA file** 29](#_Toc134141962)

[**Figure 2. 4 Database creation** 29](#_Toc134141963)

[**Figure 2. 5 V$views query** 30](#_Toc134141964)

[**Figure 2. 6 V$views query** 31](#_Toc134141965)

[**Figure 2. 7 V$views query** 31](#_Toc134141966)

[**Figure 2. 8 V$views query** 32](#_Toc134141967)

[**Figure 2. 9 Tablespaces** 32](#_Toc134141968)

[**Figure 3. 2 Tea\_owner user** 33](#_Toc134141969)

[**Figure 3. 3 Show user.** 33](#_Toc134141970)

[**Figure 3. 4 Tablespace** 34](#_Toc134141971)

[**Figure 3. 5 Tablespace backup** 34](#_Toc134141972)

[**Figure 3. 6 Tablespace query** 34](#_Toc134141973)

[**Figure 3. 7 Default tablespace** 34](#_Toc134141974)

[**Figure 3. 8 TEA\_data query** 35](#_Toc134141975)

[**Figure 3. 9 Directory** 35](#_Toc134141976)

[**Figure 3. 10 Directory grant** 35](#_Toc134141977)

[**Figure 3. 11 supplier load** 37](#_Toc134141978)

[**Figure 3. 12 supplier load query** 40](#_Toc134141979)

[**Figure 3. 13 Main supplier load** 42](#_Toc134141980)

[**Figure 3. 14 Company load** 44](#_Toc134141981)

[**Figure 3. 15 Company load query** 44](#_Toc134141982)

[**Figure 3. 16 Sale directory** 44](#_Toc134141983)

[**Figure 3. 17 Sale load** 48](#_Toc134141984)

[**Figure 3. 18 Sale load count** 48](#_Toc134141985)

[**Figure 3. 19 Product load** 49](#_Toc134141986)

[**Figure 3. 20 Product load query** 50](#_Toc134141987)

[**Figure 3. 21 Payment load** 51](#_Toc134141988)

[**Figure 3. 22 Employee load** 53](#_Toc134141989)

[**Figure 3. 23 Employee load query** 53](#_Toc134141990)

[**Figure 3. 24 All sale load** 55](#_Toc134141991)

[**Figure 3. 25 All sale load** 55](#_Toc134141992)

[**Figure 3. 26 Booking number load** 56](#_Toc134141993)

[**Figure 3. 27 Main supplier** 57](#_Toc134141994)

[**Figure 3. 28 Main supplier load** 58](#_Toc134141995)

[**Figure 3. 29 Main supplier query** 58](#_Toc134141996)

[**Figure 3. 30 Supplier table** 59](#_Toc134141997)

[**Figure 3. 31 Supplier query** 59](#_Toc134141998)

[**Figure 3. 32 Customer** 60](#_Toc134141999)

[**Figure 3. 33 Customer load** 60](#_Toc134142000)

[**Figure 3. 34 Payment type** 61](#_Toc134142001)

[**Figure 3. 35 Payment method** 62](#_Toc134142002)

[**Figure 3. 36 Cust credit card** 62](#_Toc134142003)

[**Figure 3. 37 Alter table** 63](#_Toc134142004)

[**Figure 3. 38 Cust credit card query** 63](#_Toc134142005)

[**Figure 3. 39 Class** 64](#_Toc134142006)

[**Figure 3. 40 Destination** 64](#_Toc134142007)

[**Figure 3. 41 Affiliation** 65](#_Toc134142008)

[**Figure 3. 42 Company** 66](#_Toc134142009)

[**Figure 3. 43 Product** 66](#_Toc134142010)

[**Figure 3. 44 Product query** 67](#_Toc134142011)

[**Figure 3. 45 Payment query** 68](#_Toc134142012)

[**Figure 3. 46 Commission** 69](#_Toc134142013)

[**Figure 3. 47 Commission query** 70](#_Toc134142014)

[**Figure 3. 48 Itinerary** 71](#_Toc134142015)

[**Figure 3. 49 Itinerary query** 71](#_Toc134142016)

[**Figure 3. 50 Booking number** 72](#_Toc134142017)

[**Figure 3. 51 Booking** 73](#_Toc134142018)

[**Figure 3. 52 Booking query.** 73](#_Toc134142019)

[**Figure 3. 53 Department** 73](#_Toc134142020)

[**Figure 3. 54 Insert statement.** 74](#_Toc134142021)

[**Figure 3. 55 Department query** 74](#_Toc134142022)

[**Figure 3. 56 Employee** 75](#_Toc134142023)

[**Figure 3. 57 Sale** 75](#_Toc134142024)

[**Figure 3. 58 Sale query** 76](#_Toc134142025)

[**Figure 3. 59 Tax quote** 77](#_Toc134142026)

[**Figure 3. 60 Tax quote query** 77](#_Toc134142027)

[**Figure 4. 1 Tea owner** 79](#_Toc134142028)

[**Figure 4. 2 Show user** 79](#_Toc134142029)

[**Figure 4. 3 Senior agent role** 80](#_Toc134142030)

[**Figure 4. 4 Senior agent query** 81](#_Toc134142031)

[**Figure 4. 5 Commission specialist** 81](#_Toc134142032)

[**Figure 4. 6 Commission specialist query** 82](#_Toc134142033)

[**Figure 4. 7 Intermediate agent** 83](#_Toc134142034)

[**Figure 4. 8 Intermediate agent query** 83](#_Toc134142035)

[**Figure 4. 9 Junior agent** 84](#_Toc134142036)

[**Figure 4. 10 Junior agent query** 85](#_Toc134142037)

[**Figure 4. 11 Manager role** 87](#_Toc134142038)

[**Figure 4. 12 Manager query** 87](#_Toc134142039)

[**Figure 4. 13 Grant statement** 88](#_Toc134142040)

[**Figure 4. 14 Privilege query** 88](#_Toc134142041)

[**Figure 4. 15 Create TEA users.** 90](#_Toc134142042)

[**Figure 4. 16 Dba users query** 90](#_Toc134142043)

[**Figure 4. 17 Junior agent grant** 91](#_Toc134142044)

[**Figure 4. 18 Intermediate agent grant** 92](#_Toc134142045)

[**Figure 4. 19 Senior agent grant** 92](#_Toc134142046)

[**Figure 4. 20 Commission specialist grant** 93](#_Toc134142047)

[**Figure 4. 21 Manager grant** 93](#_Toc134142048)

[**Figure 4. 22 Owner grant** 93](#_Toc134142049)

[**Figure 4. 23 Profiles for existing roles** 94](#_Toc134142050)

[**Figure 4. 24 Resource limit** 94](#_Toc134142051)

[**Figure 4. 25 Junior time profile to existing users** 95](#_Toc134142052)

[**Figure 4. 26 Intermediate time profile to existing users** 96](#_Toc134142053)

[**Figure 4. 27 Senior agent profile to existing users** 96](#_Toc134142054)

[**Figure 4. 28 Manager time profile to existing user** 97](#_Toc134142055)

[**Figure 4. 29 Owner time to existing user** 97](#_Toc134142056)

[**Figure 4. 30 Dba profile query** 97](#_Toc134142057)

[**Figure 5. 1 Listener.ora file** 98](#_Toc134142058)

[**Figure 5. 2 Lsnrctl utility** 99](#_Toc134142059)

[**Figure 5. 3 Tnsnames.ora file** 100](#_Toc134142060)

[**Figure 5. 4 Sqlnet.ora** 101](#_Toc134142061)

[**Figure 6. 1 Cold backup** 105](#_Toc134142062)

[**Figure 6. 2 Rman backup** 106](#_Toc134142063)

[**Figure 7. 1 Sample invoice** 107](#_Toc134142064)

[**Figure 7. 2 Stored procedure.** 110](#_Toc134142065)

[**Figure 7. 3 Stored procedure query.** 111](#_Toc134142066)

[**Figure 7. 4 Commission query** 112](#_Toc134142067)

[**Figure 7. 5 New customer inserted** 113](#_Toc134142068)

[**Figure 7. 6 New customer query** 113](#_Toc134142069)

List of Tables

[**Table 1. 1 ERD Documentation** 20](#_Toc134142070)

[**Table 1. 2 Customer** 21](#_Toc134142071)

[**Table 1. 3 Agent** 21](#_Toc134142072)

[**Table 1. 4 Province** 21](#_Toc134142073)

[**Table 1. 5 Country** 21](#_Toc134142074)

[**Table 1. 6 Class Type** 21](#_Toc134142075)

[**Table 1. 7 Destination** 22](#_Toc134142076)

[**Table 1. 8 Trip Type** 22](#_Toc134142077)

[**Table 1. 9 Fee** 22](#_Toc134142078)

[**Table 1. 10 Affiliation** 22](#_Toc134142079)

[**Table 1. 11 Product** 22](#_Toc134142080)

[**Table 1. 12 Supplier** 23](#_Toc134142081)

[**Table 1. 13 Supplier Affiliation** 23](#_Toc134142082)

[**Table 1. 14 TEA Office** 23](#_Toc134142083)

[**Table 1. 15 Booking** 24](#_Toc134142084)

[**Table 1. 16 Payment\_status** 24](#_Toc134142085)

[**Table 1. 17 Payment\_type** 24](#_Toc134142086)

[**Table 1. 18 Itinerary** 25](#_Toc134142087)

[**Table 1. 19 Payment** 25](#_Toc134142088)

[**Table 1. 20 Passenger** 25](#_Toc134142089)

[**Table 2. 1 Disk configuration** 26](#_Toc134142090)

# **Workshop 1**

## **Day 1 Interview questions**

1. How long has the company been in operation for?

Ans: 15years

1. How long will you use this server until the next upgrade?

Ans: 30years

1. Where do you see in your company in xxxx years? What is the yearly growth? The reason we ask is to recommend server requirement.

Ans: We see the company having about 30 employees in about 10years.

1. Do you need a server room? Is the server ready or do we have to set it up? Do we need to run cables from the server to each computer?

Ans: We already have a server room.

1. What is the start and due date of this project?

Ans: Nov 1, 2022

1. What does the daily activity your company look like?

Ans: about 8 employees are in the office, including the manager.

1. How many travel agents come to work every day?

Ans: 8

1. Are there agents working from home? The reason for this question is to know if we are going to need to set up a remote desktop connection.

Ans: Yes, there are agents who work away from the office.

1. How many transactions does your company have daily, weekly, and monthly?

Ans: about 50 weeklies

1. Does your company book ticket for large groups of people?

Ans: Yes

1. How often do you want the server to back up?

Ans: Daily

1. Do you want the database to have a function to calculate GST outside of Canada?

Ans: Not necessary.

1. If the fund is not enough, what components are required to have in database?

Ans: We would discuss that in the future.

1. You mentioned that all agents agree that if an expensive trip is booked, which results in a substantial commission, the booking fee is waived. But what’s the amount of substantial commission?

Ans: The supplier will determine it.

1. You mentioned that we could add more columns to the excel sheet. So, do you want the database to include employee information, such as salary and commissions?

Ans: Yes, it would house more information about the company.

1. In the first page you mentioned that each agent contributed some information that they currently collect to the project. What kind of method does each agent use to collect their information? Electronic or paper? Who converts?

Ans: Most agents typically use sticky notes.

1. There is one empty table about affiliation, who is going to provide the data for this table?

Ans: We would try to look for it, otherwise we leave it as unresolved.

## **Day 2 Interview questions**

1. I see there are only credit card transactions in sales in spreadsheet. Do you have any other payment type transactions?

Ans: Yes, we take cash payments.

1. Customers’ phone numbers in spreadsheet have more than 10 digits. How do you want to manage them? Will you contact the customer to get the correct number?

Ans: You can make the corrections for us or leave it as unresolved.

1. Customer ID (only 3 digits), product ID (from 2 to 4 digits), Product Category (only 3 digits).

Ans: You can recommend a better way to do this.

1. In background page 4, we see that same supplier IDs associate with same representative, but not supplier name. Could you explain?

Ans: You can recommend a better way to represent this, we did that because different suppliers can supply the same product.

1. We noticed that you have 2 region ID’s, can you tell us the region names and the reason for having two region IDs?

Ans: it is for different continents, but more can be added when needed.

1. Can you provide more details about the product ID’s that you have? Do you have a code table for it?

Ans: We would provide more details about that.

1. There are some dates errors in the sales spreadsheet. How do you want us to manage?

Ans: You can list the incorrect dates as unresolved.

1. Options for networks – we have decided to set up 3 LANs, first including all the computers for employees. The second includes server, printer, scanner and fax machines. Third is optional, that is an AP, will provide wireless network access to your mobile devices like laptops and mobile phones.

Ans: we would discuss all this in the future because it is not a priority right now.

1. Options – Include owner, the company has 18 employees. We have two options for you, a server that can manage 25 people and a server that can manage 50 people. The price difference is only about 30-50 thousand.

Ans: No answer

1. You mentioned that all agents agree that if an expensive trip is booked, which results in a substantial commission, the booking fee is waived. But what is the amount of substantial commission? Do you have a standard for it?

Ans: as said earlier, it is the suppliers who determine the commission fee. It is only our responsibility to accept this fee whether it’s reasonable for us or not.

1. Do you have the formula for commissions from different suppliers? So, our database can calculate and check if you have received correct commissions.

Ans: Like we discussed earlier, it depends on the suppliers. There is no formular.

1. Yesterday you mentioned that you want your data to be backed up daily. There are 3 different types of backups, I would like to explain the pros and cons, so you can make your choice.

Ans: You can list this for us in the recommendations table.

# **Workshop 2**

## **Entity Relationship Diagram (ERD)**

**Chart, box and whisker chart

Description automatically generated**

**Figure1. 1 ERD of Travel Expert Age**

### ERD documentation table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity** | **Description** |  | **Relationship** | **Attributes** |
| 1. Customer | Stores the attributes of each customer like name, phone number, address etc. |  | * It connects with several bridge tables to solve the many to many relationship | * Customerid: unique value for each customer * First\_name: customer first name * Last\_name: customer last name * Email: customer email * Customer\_phone\_id: unique value for customers phone types * Date\_of\_birth: customer birthdate * Address: customer address |
| 1. Customer\_Phone | It is a bridge table to resolve the many to many relationships between customer and phone types |  | * Connects with Customer on Many to one relationship * Connects to phone\_type on many to one relationship | * Customer\_id: foreign key from customer table * Phone\_type\_id: foreign key from phone type table |
| 1. Phone\_type | It stores different customer phone types e.g. mobile phone, business phone e.t.c. |  | * Connects to customer\_phone on a one to many relationship | * Phone\_type\_id: unique value for different phone types * Phone\_name: name of the phone type |
| 1. Agent | Stores attributes of each agents like first name, last name e.t.c. |  | * Connects to supplier table on a many to one relationship | * Agent\_code: unique value for each agent * First\_name: agent first name * Last\_name: agent last name |
| 1. Province | Stores province name and province code |  | * Connects to customer on a one to one relationship * Connects to supplier on a one to one relationship | * Province\_name: name of province * Province\_code: short unique value for each province e.g YYC, YYQ |
| 1. Country | Stores country name and country code |  | * connects to supplier on a one to one relationship | * Country\_name: name of country * Country\_code: unique value of each country |
| 1. Class Type | Stores details about what class customer used e.g economy class, first class e.t.c. |  | * Connects to booking on a many to one relationship | * Class\_id: unique value for each class * Class\_name: name of class used |
| 1. Destination | Stores details about the booking destination |  | * Connects to booking on a many to one relationship | * Destination\_id: unique value for each booking destination * Destination\_name: name of destination |
| 1. Trip Type | Stores details about the trip whether it’s a business trip, family, leisure e.t.c. |  | * Connects to booking on a many to one relationship | * Trip\_id: unique value of each trip * Trip\_name: name of trip booked |
| 1. Fees | Stores details about the agent fee |  | * Connects to booking on a many to one relationship | * Fee\_id: unique value for each fee * Description: description of fees * Fee\_amount: amount of each fee |
| 1. Affiliation | Stores details of affiliations i.e. groups, associations |  | * Connects to supplier affiliation on a many to one relationship | * Affiliation\_id: unique value for each affiliation * Affiliation\_name: name of different affiliation |
| 1. Product | Stores product details |  | * Connects to supplier product on a many to one relationship | * Product\_id: unique value for each product * Product description: description of different product |
| 1. Supplier | Stores details about supplier information |  | * Connects to supplier booking on a one to many relationship * Connects to agent on a one to many relationship * Connects to supplier affiliation on a one to many relationship | * Supplier\_id: unique value of supplier * Supplier\_name: name of supplier * Contact\_name: name of contact * City: supplier city * Province\_code: foreign key from province entity * Zipcode: supplier zipcode * Country\_code: foreign key from country * Phone: supplier phone number * Email: supplier email address * Website: supplier website * Fax: supplier fax number * Product\_supplier\_id: foreign key from product supplier entity * Employee\_id: foreign key from employee entity * Agent\_id: foreign key from agent entity |
| 1. Product Supplier | Bridge table between product and supplier |  | * Connects to product on a many to one relationship * Connects to supplier on a many to one relationship | * Supplier\_id: foreign key and primary key * Product\_id: foreign key and primary key * Product\_supplier\_id: unique value for each product supplier |
| 1. Supplier Affiliation | Bridge table between supplier and affiliation |  | * Connects to supplier on a many to one relationship * Connects to affiliation on a many to one relationship | * Product\_supplier\_id: foreign key and primary key * Affiliation\_id: foreign key and primary key |
| 1. TEA Office | Stores details about the Travel Agency Expert e.g. location, province e.t.c. |  | * Connects to employee on a one to many relationship | * Office\_id: unique value for each office * City: city of office * Province\_code: foreign key from province entity * Employee\_id: foreign key from employee entity |
| 1. Employee | Stores information of each employee e.g. first name, last name e.t.c. |  | * Connects to supplier a many to one relationship * Connects to tea office on a many to one relationship | * Employee\_id: unique value for each employee * First\_name: employee first name * Last\_name: employee last name * Office\_id: foreign key from TEA office entity |
| 1. Booking | Stores details about the booking e.g. booking date, amount, passengers e.t.c. |  | * Connects to destination on a one to many relationship * Connects to fee on a one to many relationship * Connects to booking payment on a one to many relationship | * Booking\_id: unique value for each booking * Booking\_date: date of booking * Customer\_id: foreign key from customer entity * Agent\_code: foreign key * Supplier\_office: office of supplier * Trip\_start: start of trip * Trip\_end: end of trip * Class\_id: foreign key from class entity * Number\_of\_travellers: number of travelers * Product\_id: foreign key from product entity * Product\_supplier: foreign key from product supplier entity * Trip\_id: foreign key from trip entity * Destination\_id: foreign key from destination entity * Fee\_id: foreign key from fee entity * Description: description of booking * Base\_price: base price of booking * Billed\_amount: booking amount including tax * Agent\_fee: fee collected by agent * Agent\_commission: commission collected by agent |
| 1. Payment Status | Stores the status of the payment if it’s a deposit or full payment |  | * Connects to payment on a many to one relationship | * Payment\_status\_id: unique value for payment status * Payment\_status\_description: description of payment status |
| 1. Payment Type | Stores the types of payment whether its cash, credit card, gift card e.t.c. |  | * Connects to payment on a many to one relationship | * Payment\_type\_id: unique value for each payment type * Payment\_description: description of payment type |
| 1. Payment | Stores details about paymemt e.g. card number, expiry e.t.c. |  | * Connects to booking payment on a one to many relationship * Connects to payment type on a many to one relationship | * Payment\_id: ubique value for each payment * Payment\_status-id: foreign key from payment status entity * Payment\_type\_id: foreign key from payment type entity * Card\_number: payment card number * Expiry: card expiry date * Billing\_date: date of payment |
| 1. Booking Payment | Bridge table between booking and payment |  | * Connects to payment on a on a many to one relationship * Connects to booking on a many to one relationship | * Payment\_id: foreign key and primary key * Booking\_id: foreign key and primary key |
| 1. Passenger | Stores details about the passenger e.g. first name, last name e.t.c. |  | * Connects to passenger itinerary on a one to many relationship | * First\_name: passenger first name * Last\_name: passenger last name * Date\_of\_birth: passenger birthdate * Email: passenger email * Passenger\_id: unique value of each passenger |
| 1. Itinerary | Stores details about itinerary e.g. start date, end date e.t.c. |  | * Connects to passenger itinerary on a one to many relationship | * Itinerary\_id: unique value for each itinerary * Customer\_id: foreign key from customer entity * Product\_name: name of product * Trip\_id: foreign key from trip entity * Class\_id: foreign key from class entity * Start\_date: start date of booking * End\_date: end date of booking |
| 1. Passenger Itinerary | Bridge table between passenger and itinerary |  | * Connects to passenger on a many to one relationship * Connects to itinerary on a many to one relationship | * Itinerary\_id: foreign key from itinerary entity * Passenger\_id: foreign key from passenger table |

**Table 1. 1 ERD Documentation**

#### **ERD table instance charts**

**Table 1. 2 Customer**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Customer\_id | PK | NN, U |  |  | Number | 5 | 001 |
| First\_name |  | NN |  |  | Varchar2 | 15 | Femi |
| Last\_name |  | NN |  |  | Varchar2 | 15 | Adams |
| Email |  |  |  |  | Varchar2 | 15 | [adams@gmail.com](mailto:adams@gmail.com) |
| Customer\_phone\_id | FK |  | Customer\_phone | Customer\_phone\_id | Number | 5 | 1 |
| Date\_of\_birth |  | NN |  |  | Date |  | 13/01/1997 |
| Address |  |  |  |  | Varchar2 | 25 | 13 main street |

**Table 1. 3 Agent**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Agent\_code | PK | NN, U |  |  | Number | 5 | BK |
| First\_name |  | NN |  |  | Varchar2 | 15 | Bryan |
| Last\_name |  | NN |  |  | Varchar2 | 15 | Kelly |

**Table 1. 4 Province**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Province\_code | PK | NN, U |  |  | Varchar2 | 5 | ON |
| Province\_name |  | NN |  |  | Varchar2 | 15 | Ontario |

**Table 1. 5 Country**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Country\_code | PK | NN, U |  |  | Varchar2 | 5 | CA |
| Country\_name |  | NN |  |  | Varchar2 | 15 | Canada |

**Table 1. 6 Class Type**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Class\_id | PK | NN, U |  |  | Number | 5 | 100 |
| Class\_name |  | NN |  |  | Varchar2 | 15 | First Class |

**Table 1. 7 Destination**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Destination\_id | PK | NN, U |  |  | Number | 5 | 010 |
| Destination\_name |  | NN |  |  | Varchar2 | 15 | Florida |

**Table 1. 8 Trip Type**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Trip\_id | PK | NN, U |  |  | Number | 5 | 110 |
| Trip\_name |  | NN |  |  | Varchar2 | 15 | Business trip |

**Table 1. 9 Fee**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Fee\_id | PK | NN, U |  |  | Number | 5 | 100 |
| Description |  | NN |  |  | Varchar2 | 25 | Holland, Germany, England |

**Table 1. 10 Affiliation**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Affiliation\_id | PK | NN, U |  |  | Varchar2 | 15 | ACTA |
| Affiliation\_name |  | NN |  |  | Varchar2 | 25 | Association of Canadian Travel Agents |

**Table 1. 11 Product**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Product\_id | PK | NN, U |  |  | Number | 5 | 150 |
| Product\_name |  | NN |  |  | Varchar2 | 15 | Cruise Line |

**Table 1. 12 Supplier**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Supplier\_id | PK | NN, U |  |  | Number | 5 | 101 |
| Contact\_name |  | NN |  |  | Varchar2 | 15 | Terry |
| Address |  |  |  |  | Varchar2 | 25 | 14, downtown street |
| City |  | NN |  |  | Varchar2 | 15 | Calgary |
| Province\_code | FK |  | Province | Province\_code | Varchar2 | 5 | YYC |
| Zipcode |  |  |  |  | Varchar2 | 7 | T3E 4F7 |
| Country\_code | FK |  | Country | Country\_code | Varchar2 | 5 | CA |
| Phone |  |  |  |  | Varchar2 | 15 | 403-271-2785 |
| Website |  | U |  |  | Varchar2 | 15 | Encorecruise.com |
| Fax |  |  |  |  | Varchar2 | 15 | 309-456-6345 |
| Product\_supplier\_id | FK |  | Product\_supplier | Product\_supplier\_id | Number | 5 | 10100 |
| Employee\_id | FK |  | Employee | Employee\_id | Number | 5 | 5100 |
| Agent\_id | FK |  | Agent | Agen\_id | Number | 5 | BK |

**Table 1. 13 Supplier Affiliation**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Supplier\_id | PK, FK | NN | Supplier | Supplier\_id | Number | 5 | 102 |
| Affiliation\_id | PK, FK | NN | Affiliation | Affiliation\_id | Varchar2 | 15 | ACTA |

**Table 1. 14 TEA Office**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Office\_id | PK | NN, U |  |  | Number | 5 | 01100 |
| City |  |  |  |  | Varchar2 | 15 | Calgary |
| Province\_code | FK |  | Province | Province\_code | Varchar2 | 5 | YYC |
| Employee\_id | FK | NN | Employee | Employee\_id | Number | 5 | 51001 |

**Table 1. 15 Booking**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Booking\_id | PK | NN, U |  |  | Number | 5 | 78900 |
| Booking\_date |  | NN |  |  | Date |  | 16/11/2022 |
| Customer\_id | FK | NN | Customer | Customer\_id | Number | 5 | 001 |
| Agent\_code | FK | NN | Agent | Agent\_code | Varchar2 | 5 | BK |
| Supplier\_office |  |  |  |  | Varchar2 | 15 | 23, ends road |
| Trip\_start |  | NN |  |  | Date |  | 16/11/2022 |
| Trip\_end |  | NN |  |  | Date |  | 25/12/2022 |
| Class\_id | FK | NN | Class | Class\_id | Number | 5 | 101 |
| Number\_of\_travellers |  | NN |  |  | Number | 5 | 4 |
| Product\_id | FK | NN | Product | Product\_id | Number | 5 | 151 |
| Product\_supplier\_id | FK | NN | Product\_supplier | Product\_supplier\_id | number | 5 | 2001 |
| Trip\_id | FK | NN | Trip | Trip\_id | Number | 5 | 115 |
| Destination\_id | FK | NN | Destination | Destination\_id | Number | 5 | 020 |
| Fee\_id | FK | NN | Fee | Fee\_id | Number | 5 | 150 |
| Description |  |  |  |  | Varchar2 | 15 | Holland, Germany |
| Base\_price |  | NN |  |  | Number | 6,2 | 2000.00 |
| Billed\_amount |  | NN |  |  | Number | 6,2 | 2300.45 |
| Agent\_fee |  | NN |  |  | Number | 6,2 | 50.00 |
| Agent\_commission |  | NN |  |  | Number | 6,2 | 210.00 |

**Table 1. 16 Payment\_status**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Payment\_status\_id | PK | NN, U |  |  | Number | 5 | 7001 |
| Payment\_status\_description |  | NN |  |  | Varchar2 | 25 | Deposit |

**Table 1. 17 Payment\_type**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Payment\_type\_id | PK | NN, U |  |  | Number | 5 | 008 |
| Payment\_description |  | NN |  |  | Varchar2 | 15 | Credit card |

**Table 1. 18 Payment**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Payment\_id | PK | NN, U |  |  | Number | 5 | 20082 |
| Payment\_status\_id | FK | NN | Payment\_status | Payment\_status\_id | Number | 5 | 7002 |
| Payment\_type\_id | FK | NN | Payment\_type | Payment\_type\_id | Number | 5 | 009 |
| Card\_number |  | NN, U |  |  | Number | 5 | 789012341765 |
| Expiry |  | NN |  |  | Date |  | 17/11/2022 |
| Billing\_date |  | NN |  |  | Date |  | 16/11/2022 |

**Table 1. 18 Itinerary**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Itinerary\_id | PK | NN, U |  |  | Number | 5 | 7001 |
| Customer\_id | FK | NN | Customer | Customer\_id | Number | 5 | 010 |
| Product\_name |  | NN |  |  | Varchar2 | 15 | Cruise Line |
| Trip\_id | FK | NN | Trip | Trip\_id | Number | 5 | 111 |
| Class\_id | FK | NN | Class | Class\_id | Number | 5 | 003 |
| Start\_date |  | NN |  |  | Date |  | 28/11/2022 |
| End\_date |  | NN |  |  | Date |  | 17/12/2022 |

**Table 1. 19 Payment**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Payment\_id | PK, FK | NN | Payment | Payment\_id | Number | 5 | 26780 |
| Booking\_id | PK, FK | NN | Booking | Booking\_id | Number | 5 | 67098 |

**Table 1. 20 Passenger**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column Name | Key Type | Nulls/  Unique | FK Ref Table | FK Ref Columns | Data Type | Maximum Length | Sample Data 1 |
| Passenger\_id | PK | NN, U |  |  | Number | 5 | 0087 |
| First\_name |  | NN |  |  | Varchar2 | 15 | Nkem |
| Last\_name |  | NN |  |  | Varchar2 | 15 | Mba |
| Date\_of\_birth |  | NN |  |  | Date |  | 13/12/1985 |
| Email |  |  |  |  | Varchar2 | 15 | [nkem@gmail.com](mailto:nkem@gmail.com) |

# **Workshop 3**

## **Disk configuration blueprint**

|  |  |
| --- | --- |
| **DISK NAME** | **DESCRIPTION** |
| Disk C | Operating System  InitTEA  Pfile  Backups  Temp\createTEA.sql |
| Disk D | User Database Data Files  Controlfile  Datafile  Onlinelog |
| Disk E | User Database Log Files  Controlfile  Datafile  Onlinelog |
| Disk F; | System Databases  Controlfile  Datafile  Onlinelog |

**Table 2. 1 Disk configuration**

### Steps to create the database.

**Step 1**

Create init.ora file which has 3 control files stored on different disk space.

C:\Oracle\db\19C\database\initTEA.ora is the directory path on Windows OS where the file is stored

db\_name= TEA

control\_files='D:\Oracle\Oradata\TEA\controlfile\Control01.ctl','E:\Oracle\Oradata\TEA\controlfile\Control02.ctl', 'F:\Oracle\Oradata\TEA\controlfile\Control03.ctl'

db\_block\_size=8192

db\_domain=femi.saitclass.com

memory\_target=2g

pga\_aggregate\_target=1g

undo\_tablespace=UNDOTS

**Graphical user interface, text, application

Description automatically generated**

**Figure 2. 1 Init.ora file**

Graphical user interface, text, application

Description automatically generated

**Figure 2. 2 Init.ora directory**

**Step 2**

Create the Windows Service for the database instance by running this script in sqlplus

TEA- stands for Travel Expert Agency

Pfile is the same as init.ora file

oradim -new -sid TEA -pfile c:\Oracle\db\19C\database\initTEA.ora

startup nomount

**Step 3:**

Start up the new database instance in no mount stage.

Run startup nomount in sqlplus

Hint: You need to connect to the database as SYSTEM before running this script

**Step 4:**

Write the code that specifies the parameters the database must follow.

C:\Temp\createTEA.sql is the code that creates the database, and it is stored in temp directory on C drive

CREATE DATABASE TEA

USER SYS IDENTIFIED BY Password1

USER SYSTEM IDENTIFIED BY Password1

LOGFILE GROUP 1 ('d:\Oracle\oradata\TEA\onlinelog\redo01a.log') SIZE 10M BLOCKSIZE 512,

GROUP 2 ('e:\Oracle\oradata\TEA\onlinelog\redo02a.log') SIZE 10M BLOCKSIZE 512,

GROUP 3 ('f:\Oracle\oradata\TEA\onlinelog\redo03a.log') SIZE 10M BLOCKSIZE 512

MAXLOGFILES 5

MAXLOGMEMBERS 5

MAXLOGHISTORY 1

MAXDATAFILES 100

MAXINSTANCES 1

CHARACTER SET US7ASCII

NATIONAL CHARACTER SET AL16UTF16

EXTENT MANAGEMENT LOCAL

DATAFILE 'd:\Oracle\oradata\TEA\datafile\system01.dbf' SIZE 500M

SYSAUX DATAFILE 'e:\Oracle\oradata\TEA\datafile\sysaux01.dbf' SIZE 500M

DEFAULT TABLESPACE users DATAFILE 'f:\Oracle\oradata\TEA\datafile\users01.dbf' SIZE 500M

DEFAULT TEMPORARY TABLESPACE tempts TEMPFILE 'd:\Oracle\oradata\TEA\datafile\temp01.dbf' SIZE 200M

UNDO TABLESPACE undots DATAFILE 'e:\Oracle\oradata\TEA\datafile\undotbs01.dbf' SIZE 200M;

Graphical user interface, text

Description automatically generated

**Figure 2. 3 Create TEA file**

**Step 5:**

Create the database by running this script in sqlplus.

@c:\Temp\createTEA.sql

Text

Description automatically generated

**Figure 2. 4 Database creation**

**Step 6:**

Run a script called catalog.sql in sqlplus.

The path for this script is C:\Oracle\db\19c\rdbms\admin

Purpose: it creates the views of the data dictionary tables, the dynamic performance views, and public synonyms for many of the views. Grants PUBLIC access to the synonyms.

@c:\Oracle\db\19C\rdbms\admin\catalog.sql

**Step 7:**

Run a script called catproc.sql in sqlplus.

The path for this script is C:\Oracle\db\19c\rdbms\admin

Purpose: runs all scripts required for or used with PLSQL.

@c:\Oracle\db\19C\rdbms\admin\catproc.sql

**Step 8:**

Run a script called pupbld.sql in sqlplus

The path for this script is C:\Oracle\db\19c\sqlplus\admin

Purpose: Script to install the SQL\*Plus PRODUCT\_USER\_PROFILE tables.

@c:\Oracle\db\19C\sqlplus\admin\pupbld.sql

**Step 9:**

Run a SQL query to show v$controlfile and block size in sqlplus

Purpose: this displays the names of the control files.

select name from v$controlfile;

select block\_size from v$controlfile;Text

Description automatically generated

**Figure 2. 5 V$views query**

Text

Description automatically generated

**Figure 2. 6 V$views query**

**Step 10:**

Run a sql query to show v$datalfile in sqlplus

Purpose: this displays datafile information from the control files.

select name from v$datafile;

Text

Description automatically generated

**Figure 2. 7 V$views query**

**Step 11:**

Run a sql query to show v$logfile in sqlplus

Purpose: it contains information about red log files.

desc v$logfile

**Text

Description automatically generated**

**Figure 2. 8 V$views query**

**Step 12:**

Run a sql query to show tablespaces in sqlplus

Purpose: tablespace is where logical data is stored in the database.

Select tablespace\_name from dba\_tablespaces;

**Text

Description automatically generated**

**Figure 2. 9 Tablespaces**

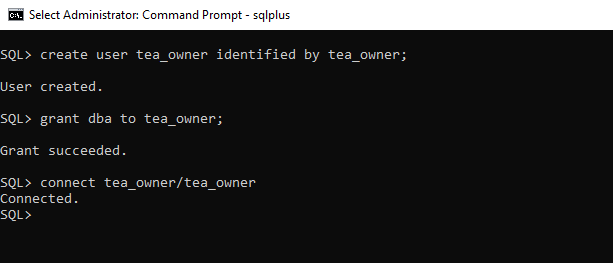
# **Workshop 4**

## **DBA User Creation**

SQL> create user tea\_owner identified by tea\_owner;

SQL> grant dba to tea\_owner;

SQL> connect tea\_owner/tea\_owner



**Figure 3. 2 Tea\_owner user**

SQL> show user

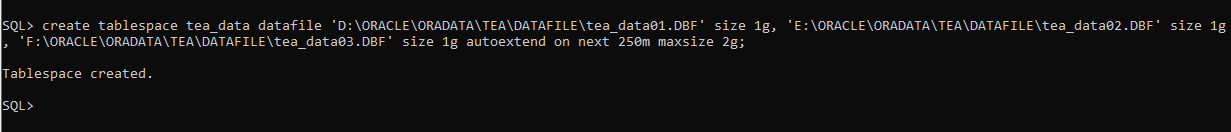
Text

Description automatically generated

**Figure 3. 3 Show user.**

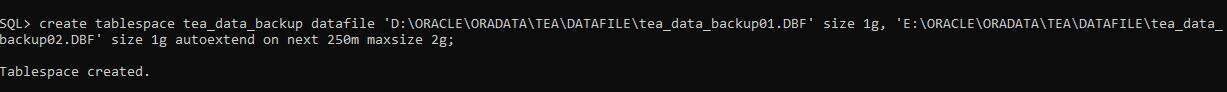
### Tablespace creation

SQL> create tablespace tea\_data datafile 'D:\ORACLE\ORADATA\TEA\DATAFILE\tea\_data01.DBF' size 1g, 'E:\ORACLE\ORADATA\TEA\DATAFILE\tea\_data02.DBF' size 1g, 'F:\ORACLE\ORADATA\TEA\DATAFILE\tea\_data03.DBF' size 1g autoextend on next 250m maxsize 2g;



**Figure 3. 4 Tablespace**

create tablespace tea\_data\_backup datafile 'D:\ORACLE\ORADATA\TEA\DATAFILE\tea\_data\_backup01.DBF' size 1g, 'E:\ORACLE\ORADATA\TEA\DATAFILE\tea\_data\_backup02.DBF' size 1g autoextend on next 250m maxsize 2g;



**Figure 3. 5 Tablespace backup**

select tablespace\_name from dba\_tablespaces;

Graphical user interface, text

Description automatically generated

**Figure 3. 6 Tablespace query**

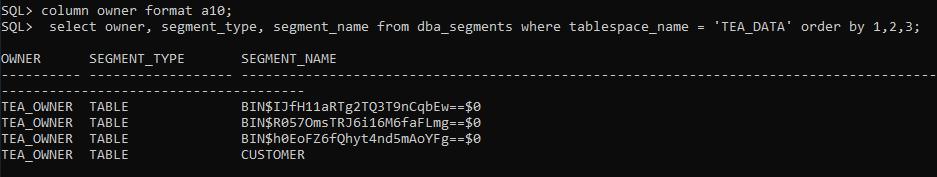
SQL> alter database default tablespace tea\_data;

Text

Description automatically generated

**Figure 3. 7 Default tablespace**

SQL> select owner, segment\_type, segment\_name from dba\_segments where tablespace\_name = 'TEA\_DATA' order by 1,2,3;



**Figure 3. 8 TEA\_data query**

#### **External table creation**

**Customer load external table**

Lookup table on original excel data

SQL> create directory tea\_customer as 'C:\Temp\Loader';



**Figure 3. 9 Directory**

SQL> grant read, write on directory tea\_customer to tea\_owner;

Text

Description automatically generated

**Figure 3. 10 Directory grant**

SQL> @C:\connTemp\Loader\teacustomertable.sql

create table customer\_load

(

customer\_id number(6),

first\_name varchar2(50),

last\_name varchar2(50),

agent\_code varchar2(3),

email varchar2(50),

home\_phone varchar2(15),

business\_phone varchar2(15),

date\_of\_birth date,

address varchar2(50),

city varchar2(25),

postal\_code varchar2(7),

province varchar2(7),

country varchar2(15)

)

organization external

(

type oracle\_loader

default directory tea\_customer

access parameters

(

records delimited by newline

fields terminated by ","

optionally enclosed by '"'

(

customer\_id CHAR,

first\_name CHAR,

last\_name CHAR,

agent\_code CHAR,

email CHAR,

home\_phone CHAR,

business\_phone CHAR,

date\_of\_birth date “mm/dd/yyyy”,

address CHAR,

city CHAR,

postal\_code CHAR,

province CHAR,

country CHAR

)

)

location ('teacustomer.csv')

)

/

**Supplier load external table**

This table is a look up table for the original data in the excel sheet

SQL> create directory tea\_supplier as 'C:\Temp\Loader';

SQL> select directory\_name, directory\_path, origin\_con\_id from dba\_directories;Text

Description automatically generated

**Figure 3. 11 supplier load**

SQL> @C:\Temp\Loader\teasuppliertable.sql

Create table supplier load file

create table supplier\_load

(

product\_supplier number(6),

product\_category number(6),

supplier\_office number(1),

product\_description varchar2(50),

contact\_name varchar2(50),

company varchar2(80),

address1 varchar2(50),

address2 varchar2(50),

city varchar2(25),

province varchar2(2),

postal\_code varchar2(7),

country varchar2(15),

phone\_number number(12),

fax number(12),

email varchar2(50),

website varchar2(50),

represents varchar2(25),

affiliation varchar2(25)

)

organization external

(

type oracle\_loader

default directory tea\_supplier

access parameters

(

records delimited by newline

fields terminated by ","

(

product\_supplier CHAR,

product\_category CHAR,

supplier\_office CHAR,

product\_description CHAR,

contact\_name CHAR,

company CHAR,

address1 CHAR,

address2 CHAR,

city CHAR,

province CHAR,

postal\_code CHAR,

country CHAR,

phone\_number CHAR,

fax CHAR,

email CHAR,

website CHAR,

represents CHAR,

affiliation CHAR

)

)

location ('teasupplier.csv')

)

/

SQL> select \* from supplier\_load;

Text

Description automatically generated

**Figure 3. 12 supplier load query**

**Main Supplier external table**

This is a table for the suppliers that represents other suppliers

SQL> @C:\Temp\Loader\teamainsuppliertable.sql

Create main supplier load file

Main\_Supplier\_Load Table

create table main\_supplier\_load

(

product\_supplier number(6),

product\_category number(6),

supplier\_office number(1),

product\_description varchar2(50),

contact\_name varchar2(50),

company varchar2(80),

address1 varchar2(50),

address2 varchar2(50),

city varchar2(25),

province varchar2(2),

postal\_code varchar2(7),

country varchar2(15),

phone\_number number(12),

fax number(12),

email varchar2(50),

website varchar2(50),

represents varchar2(25),

affiliation varchar2(25)

)

organization external

(

type oracle\_loader

default directory tea\_supplier

access parameters

(

records delimited by newline

fields terminated by ","

(

product\_supplier CHAR,

product\_category CHAR,

supplier\_office CHAR,

product\_description CHAR,

contact\_name CHAR,

company CHAR,

address1 CHAR,

address2 CHAR,

city CHAR,

province CHAR,

postal\_code CHAR,

country CHAR,

phone\_number CHAR,

fax CHAR,

email CHAR,

website CHAR,

represents CHAR,

affiliation CHAR

)

)

location ('teamainsupplier.csv')

)

REJECT LIMIT UNLIMITED;

Text

Description automatically generated

**Figure 3. 13 Main supplier load**

**Company external table**

This table shows a list of all the suppliers, their names, the supplier\_id that represents them and the supplier group they belong to. Company\_id is a unique value that is not in the original data, it represents a value of all suppliers in the table. Supplier\_rep is a foreign key of company\_id in the same table. Supplier\_rep is the supplier\_id that other suppliers belong to.

SQL> @C:\Temp\Loader\teacompany.sql

Create company load file

create table company\_load

(

company\_id number(6),

company\_name varchar2(80),

supplier\_id number(6),

supplier\_rep number(6)

)

organization external

(

type oracle\_loader

default directory tea\_supplier

access parameters

(

records delimited by newline

fields terminated by ","

OPTIONALLY ENCLOSED BY '"'

(

company\_id CHAR,

company\_name CHAR,

supplier\_id CHAR,

supplier\_rep CHAR

)

)

location ('teacompany.csv')

)

REJECT LIMIT UNLIMITED;

Text

Description automatically generated

**Figure 3. 14 Company load**

SQL> select \* from company\_load;

Text

Description automatically generated

**Figure 3. 15 Company load query**

**Sale Load external table**

Lookup table on original excel data.

SQL> create directory tea\_sale as 'C:\Temp\Loader';

Text

Description automatically generated

**Figure 3. 16 Sale directory**

SQL> @ C:\Temp\Loader\teasaletable.sql

Create sale load file

/\*=====Load Sale External Data=====\*/

CREATE TABLE sale\_load

(

sale\_date date,

customer\_id number(8),

itinerary\_number number(8),

agent\_code varchar2(3),

booking\_number varchar2(15),

product\_category number(3),

product\_supplier number(8),

supplier\_office number(1),

trip\_start date,

trip\_end date,

class varchar2(5),

number\_of\_traveller number(3),

product varchar2(25),

description varchar2(60),

destination varchar2(25),

destination\_id varchar2(5),

credit\_card varchar2(15),

expiry\_date date,

card\_number number(20),

billing\_date date,

bill\_description varchar2(15),

base\_price number(11,2),

total\_price number(11,2),

billed\_amount number(11,2),

agency\_fee\_code varchar2(3),

agency\_fee\_amount number(6,2),

agency\_commission number(8,2)

)

ORGANIZATION EXTERNAL

(

TYPE ORACLE\_LOADER

DEFAULT DIRECTORY tea\_sale

ACCESS PARAMETERS

(

RECORDS DELIMITED BY NEWLINE

FIELDS TERMINATED BY ","

OPTIONALLY ENCLOSED BY '"'

(

sale\_date date "mm/dd/yyyy",

customer\_id char,

itinerary\_number char,

agent\_code char,

booking\_number char,

product\_category char,

product\_supplier char,

supplier\_office char,

trip\_start date "mm/dd/yyyy",

trip\_end date "mm/dd/yyyy",

class char,

number\_of\_traveller char,

product char,

description char,

destination char,

destination\_id char,

credit\_card char,

expiry\_date date "mm/dd/yyyy",

card\_number char,

billing\_date date "mm/dd/yyyy",

bill\_description char,

base\_price char,

total\_price char,

billed\_amount char,

agency\_fee\_code char,

agency\_fee\_amount char,

agency\_commission char

)

)

LOCATION ('teasale.csv')

)

REJECT LIMIT UNLIMITED;

SQL> select \* from sale\_load;

Text

Description automatically generated with low confidence

**Figure 3. 17 Sale load**

SQL> select count(\*) from sale\_load;

Text

Description automatically generated

**Figure 3. 18 Sale load count**

**Product External Table**

SQL> @C:\Temp\Loader\teaproducttable.sql

Create product load file

/\*=====Load Product External Data=====\*/

CREATE TABLE product\_load

(

product\_id number(6),

product\_name varchar2(50)

)

ORGANIZATION EXTERNAL

(

TYPE ORACLE\_LOADER

DEFAULT DIRECTORY tea\_sale

ACCESS PARAMETERS

(

RECORDS DELIMITED BY NEWLINE

FIELDS TERMINATED BY ","

OPTIONALLY ENCLOSED BY '"'

(

product\_id char,

product\_name char

)

)

LOCATION ('teaproduct.csv')

)

REJECT LIMIT UNLIMITED;

Text

Description automatically generated

**Figure 3. 19 Product load**

SQL> select \* from product\_load;

Graphical user interface, text

Description automatically generated

**Figure 3. 20 Product load query**

**Payment External Table**

SQL> @C:\Temp\Loader\teapaymenttable.sql

Create payment load file

/\*=====Load Product External Data=====\*/

CREATE TABLE payment\_load

(

payment\_id number(6),

cust\_credit\_card\_id number(6),

payment\_type varchar2(15)

)

ORGANIZATION EXTERNAL

(

TYPE ORACLE\_LOADER

DEFAULT DIRECTORY tea\_sale

ACCESS PARAMETERS

(

RECORDS DELIMITED BY NEWLINE

FIELDS TERMINATED BY ","

OPTIONALLY ENCLOSED BY '"'

(

payment\_id char,

cust\_credit\_card\_id char,

payment\_type char

)

)

LOCATION ('teapayment.csv')

)

REJECT LIMIT UNLIMITED;

Text

Description automatically generated with medium confidence

**Figure 3. 21 Payment load**

**Employee external table**

SQL> @C:\Temp\Loader\teaemployeetable.sql

Create employee load file

create table employee\_load

(

employee\_id number(6),

first\_name varchar2(50),

last\_name varchar2(50),

department\_id number(3),

address varchar2(256),

city varchar2(50),

postal\_code varchar2(10),

phone\_number varchar(16),

email varchar2(50)

)

organization external

(

type oracle\_loader

default directory tea\_customer

access parameters

(

records delimited by newline

fields terminated by ","

optionally enclosed by '"'

(

employee\_id char,

first\_name char,

last\_name char,

department\_id char,

address char,

city char,

postal\_code char,

phone\_number char,

email char

)

)

location ('teaemployee.csv')

)

REJECT LIMIT UNLIMITED;

Text

Description automatically generated

**Figure 3. 22 Employee load**

SQL> select employee\_id, first\_name, last\_name, city from employee\_load;

Text

Description automatically generated

**Figure 3. 23 Employee load query**

**All Sale external table**

SQL> @C:\Temp\Loader\teaallsaletable.sql

Create all sale load file

/\*=====Load All Sale External Data=====\*/

CREATE TABLE all\_sale\_load

(

sale\_id number(6),

employee\_id number(6),

customer\_id number(6),

itinerary\_number number(8),

product\_id number(6),

class varchar2(5),

supplier\_id number(6),

payment\_id number(6),

booking\_number varchar2(15),

destination\_id varchar2(5),

sale\_date date

)

ORGANIZATION EXTERNAL

(

TYPE ORACLE\_LOADER

DEFAULT DIRECTORY tea\_sale

ACCESS PARAMETERS

(

RECORDS DELIMITED BY NEWLINE

FIELDS TERMINATED BY ","

OPTIONALLY ENCLOSED BY '"'

(

sale\_id char,

employee\_id char,

customer\_id char,

itinerary\_number char,

product\_id char,

class char,

supplier\_id char,

payment\_id char,

booking\_number char,

destination\_id char,

sale\_date date "mm/dd/yyyy"

)

)

LOCATION ('teaallsale.csv')

)

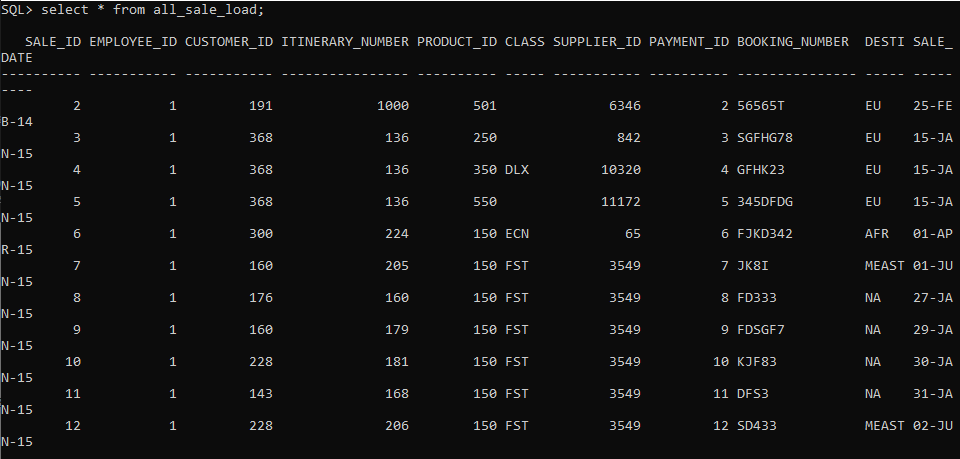
REJECT LIMIT UNLIMITED;

Text

Description automatically generated

**Figure 3. 24 All sale load**

SQL> select \* from all\_sale\_load;



**Figure 3. 25 All sale load**

**Booking number external able**

/\*=====Load Booking Number External Data=====\*/

CREATE TABLE booking\_number\_load

(

booking\_number varchar2(15)

)

ORGANIZATION EXTERNAL

(

TYPE ORACLE\_LOADER

DEFAULT DIRECTORY tea\_load

ACCESS PARAMETERS

(

RECORDS DELIMITED BY NEWLINE

FIELDS TERMINATED BY ","

OPTIONALLY ENCLOSED BY '"'

(

booking\_number char

)

)

LOCATION ('teabookingnumber.csv')

)

REJECT LIMIT UNLIMITED;

SQL> @C:\Temp\Loader\teabookingnumber.sql

Text

Description automatically generated

**Figure 3. 26 Booking number load**

**Itinerary number external table**

/\*=====Load Itinerary Number External Data=====\*/

CREATE TABLE itinerary\_number\_load

(

itinerary\_number number

)

ORGANIZATION EXTERNAL

(

TYPE ORACLE\_LOADER

DEFAULT DIRECTORY tea\_load

ACCESS PARAMETERS

(

RECORDS DELIMITED BY NEWLINE

FIELDS TERMINATED BY ","

OPTIONALLY ENCLOSED BY '"'

(

itinerary\_number char

)

)

LOCATION ('teaitinerarynumber.csv')

)

REJECT LIMIT UNLIMITED;

SQL> @C:\Temp\Loader\teaitinerarynumber.sql

Text

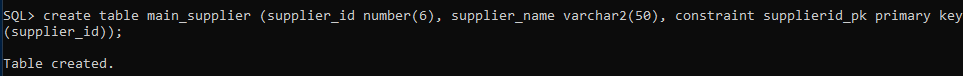
Description automatically generated

##### **Table Creation**

**Main supplier table**

This is a table for the suppliers that represents other suppliers

SQL> create table main\_supplier (supplier\_id number(6), supplier\_name varchar2(50), constraint supplierid\_pk primary key(supplier\_id));



**Figure 3. 27 Main supplier**

SQL> insert into main\_supplier select product\_supplier, company from main\_supplier\_load;

Text

Description automatically generated with medium confidence

**Figure 3. 28 Main supplier load**

SQL> select \* from main\_supplier;

Text

Description automatically generated

**Figure 3. 29 Main supplier query**

**Supplier table**

SQL> create table supplier (supplier\_id number(6), product\_category number(6), supplier\_office number(1), product\_description varchar2(50),contact\_name varchar2(50), company varchar2(80), address1 varchar2(50), address2 varchar(50), city varchar2(25), province varchar2(2), postal\_code varchar2(7), country varchar2(15), phone\_number number(12), fax number(12), email varchar2(50), website varchar2(50), represents varchar2(25), affiliation\_code varchar2(25));

SQL> insert into supplier select product\_supplier, product\_category, supplier\_office, product\_description, contact\_name,company, address1, address2, city, province, postal\_code, country, phone\_number, fax, email, website, represents, affiliation from main\_supplier\_load;

SQL> alter table supplier add (constraint supplierid\_pk1 primary key(supplier\_id), constraint supplier\_supplierid\_fk foreign key(supplier\_id) references main\_supplier(supplier\_id), constraint affiliation\_affiliation\_code\_fk foreign key(affiliation\_code) references affiliation(affiliation\_code));

Text

Description automatically generated

**Figure 3. 30 Supplier table**

SQL> select supplier\_id, company, product\_category, contact\_name, city, province from supplier;

Graphical user interface, text, application

Description automatically generated

**Figure 3. 31 Supplier query**

**Customer table**

SQL> create table Customer (customer\_id number(6), first\_name varchar2(50), last\_name varchar2(50), agent\_code varchar2(3), email varchar2(50), home\_phone varchar2(15), business\_phone varchar2(15), date\_of\_birth date, address varchar2(50), city varchar2(25), postal\_code varchar2(7), province varchar2(3), country varchar2(15)) tablespace tea\_data;

Text

Description automatically generated

**Figure 3. 32 Customer**

SQL> alter table customer add constraint customer\_id\_pk primary key(customer\_id);

SQL> insert into customer select customer\_id, first\_name, last\_name, agent\_code, email, home\_phone, business\_phone, date\_of\_birth, address, city, postal\_code, province, country from customer\_load;

SQL> select \* from customer;

SQL> select count(\*) from customer;

A picture containing text

Description automatically generated

**Figure 3. 33 Customer load**

**Payment type table**

SQL> create table payment\_type (payment\_type varchar2(15) not null);

SQL> insert into payment\_type select distinct bill\_description from sale\_load where bill\_description is not null;

SQL> alter table payment\_type add payment\_type\_id integer generated by default on null as identity;

SQL> alter table payment\_type add constraint payment\_type\_pk primary key(payment\_type);

SQL> select \* from payment\_type;

Text

Description automatically generated

**Figure 3. 34 Payment type**

**Payment method table**

SQL> create table payment\_method (payment\_method varchar2(15));

SQL> insert into payment\_method values ('cash');

SQL> insert into payment\_method values ('cheque');

SQL> insert into payment\_method values ('credit card');

SQL> insert into payment\_method values ('bank deposit');

SQL> alter table payment\_method add payment\_method\_id integer generated by default on null as identity;

SQL> alter table payment\_method add constraint payment\_methodid\_pk primary key(payment\_method\_id);

SQL> select \* from payment\_method;

Text

Description automatically generated

**Figure 3. 35 Payment method**

**Cust credit card table**

SQL> create table cust\_credit\_card (customer\_id number(6), card\_type varchar2(15), card\_number number(16), expiry\_date date);

SQL> insert into cust\_credit\_card select customer\_id, credit\_card, card\_number, expiry\_date from sale\_load;

SQL> alter table cust\_credit\_card add constraint customer\_customerid\_fk1 foreign key(customer\_id) references customer(customer\_id);

SQL> alter table cust\_credit\_card add cust\_credit\_card\_id integer generated by default on null as identity;

SQL> alter table cust\_credit\_card add constraint cust\_credit\_cardid\_pk primary key(cust\_credit\_card\_id);



**Figure 3. 36 Cust credit card**

A screenshot of a computer

Description automatically generated with medium confidence

**Figure 3. 37 Alter table**

Select \* from cust\_credit\_card;

Graphical user interface, text

Description automatically generated

**Figure 3. 38 Cust credit card query**

**Class table**

SQL> create table class (class\_type varchar2(4), constraint class\_class\_type\_pk1 primary key (class\_type));

SQL> insert into class select distinct class from sale\_load where class is not null;

SQL> select \* from class;

Text

Description automatically generated

**Figure 3. 39 Class**

**Destination table**

SQL> create table destination (destination\_id varchar(5), constraint destination\_destinationid\_pk primary key(destination\_id));

SQL> insert into destination select distinct destination\_id from sale\_load;

SQL> select \* from destination;

Text

Description automatically generated

**Figure 3. 40 Destination**

**Affiliation table**

SQL> create table affiliation (affiliation\_code varchar2(25), constraint affiliation\_affiliationcode\_pk primary key(affiliation\_code));

SQL> insert into affiliation select distinct affiliation from supplier\_load where affiliation is not null;

SQL> select \* from affiliation;

Text

Description automatically generated

**Figure 3. 41 Affiliation**

**Company table**

SQL> create table company (company\_id number(6), company\_name varchar2(80), supplier\_id number(6), supplier\_rep number(6));

SQL> insert into company select company\_id, company\_name, supplier\_id, supplier\_rep from company\_load;

SQL> alter table company add constraint companyid\_pk primary key(company\_id);

SQL> alter table company add constraint main\_supplier\_supplierid\_fk foreign key(supplier\_id) references main\_supplier(supplier\_id);

SQL> alter table company add constraint company\_supplier\_rep\_fk foreign key(supplier\_rep) references company(company\_id);

SQL> select \* from company;

Text

Description automatically generated

**Figure 3. 42 Company**

**Product table**

SQL> create table product (product\_id number(6), product\_name varchar2(50), constraint productid\_pk primary key(product\_id));

SQL> insert into product select product\_id, product\_name from product\_load;

insert into product (product\_id) values (35);

A screenshot of a computer screen

Description automatically generated with medium confidence

**Figure 3. 43 Product**

SQL> select \* from product;

Graphical user interface, text

Description automatically generated

**Figure 3. 44 Product query**

**Payment table**

SQL> create table payment (payment\_id integer generated by default on null as identity, cust\_credit\_card\_id number(6), payment\_type varchar2(15));

SQL> insert into payment select payment\_id, cust\_credit\_card\_id, payment\_type from payment\_load;

SQL> alter table payment add (constraint paymentid\_pk primary key(payment\_id), constraint payment\_type\_payment\_type\_fk foreign key(payment\_type) references payment\_type(payment\_type));

insert into payment values (591, 591,'');

insert into payment values (3,3,'');

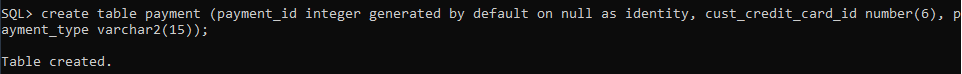
insert into payment values (4,4,'');

insert into payment values (369,369,'');

insert into payment values (510,510,'');

alter table payment add payment\_method\_id number;

alter table payment add constraint payment\_payment\_method\_id\_fk foreign key (payment\_method\_id) references payment\_method(payment\_method\_id);



Text

Description automatically generated

Graphical user interface, text

Description automatically generated

**Figure 3. 45 Payment query**

**Commission table**

Did not add FK on supplier\_id because it violates parent key. This is due to the data assigned to us, originally meant to have FK.

SQL> create table commission (amount number(6,2), supplier\_id number(6), commission\_date date, customer\_id number(6), booking\_number varchar2(15));

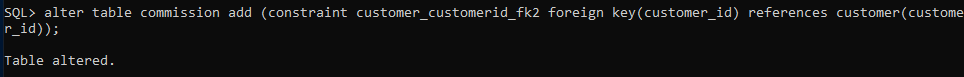
SQL> insert into commission select distinct agency\_commission, product\_supplier, trip\_end, customer\_id, booking\_number from sale\_load;

SQL> alter table commission add (commission\_id integer generated by default on null as identity);

SQL> alter table commission add constraint commissionid\_pk primary key(commission\_id);

SQL> alter table commission add (constraint customer\_customerid\_fk2 foreign key(customer\_id) references customer(customer\_id));

Text

Description automatically generated with medium confidence 

**Figure 3. 46 Commission**

SQL> select \* from commission;

Graphical user interface, text

Description automatically generated with medium confidence

**Figure 3. 47 Commission query**

**Itinerary number table**

create table itinerary\_number (itinerary\_number number);

alter table itinerary\_number add constraint itinerary\_number\_pk primary key (itinerary\_number);

**Itinerary table**

SQL> create table itinerary (itinerary\_number number(8), customer\_id number(6), trip\_start date, trip\_end date, number\_of\_travellers number(3), sale\_date date);

SQL> insert into itinerary select itinerary\_number, customer\_id, trip\_start, trip\_end, number\_of\_traveller, sale\_date from sale\_load;

SQL> alter table itinerary add constraint customer\_customerid\_fk3 foreign key(customer\_id) references customer(customer\_id);

Not Needed [delete from itinerary where rowid not in (select max(rowid) from itinerary group by itinerary\_number);] used to delete duplicate rows.

SQL> alter table itinerary add constraint itinerarynumber\_itinerary\_number\_fk foreign key(itinerary\_number) references itinerary\_number(itinerary\_number);

A picture containing text

Description automatically generated

**Figure 3. 48 Itinerary**

SQL> select \* from itinerary order by itinerary\_number;

Graphical user interface, text

Description automatically generated

**Figure 3. 49 Itinerary query**

**Booking number table**

create table booking\_number (booking\_number varchar(256));

alter table booking\_number add constraint booking\_number\_pk primary key(booking\_number);

insert into booking\_number select booking\_number from booking\_number\_load;

Text

Description automatically generated

**Figure 3. 50 Booking number**

**Booking table**

SQL> create table booking (booking\_number varchar2(15), itinerary\_number number(8), product\_id number(3), payment\_id number(6), billing\_date date, base\_price number(11,2), billed\_amount number(11,2), total\_price number(11,2), bill\_description varchar2(15), agency\_fee\_amount number(6,2), agency\_fee\_code varchar2(3));

SQL> insert into booking select distinct booking\_number, itinerary\_number, product\_id, payment\_id, billing\_date, base\_price, billed\_amount, total\_price, bill\_description, agency\_fee\_amount, agency\_fee\_code from booking\_load;

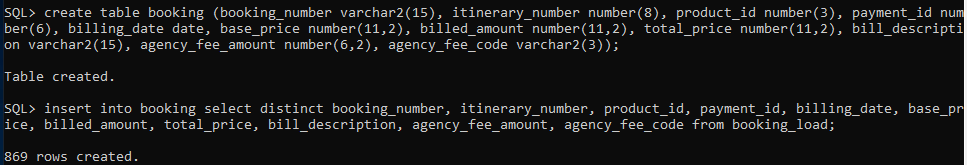
Not needed [SQL> delete from booking where rowid not in (select max(rowid) from booking group by booking\_number); (used to delete duplicate rows)]

SQL> alter table booking add constraint booking\_number\_bookingnumber\_fk foreign key (booking\_number) references booking\_number(booking\_number);

SQL> alter table booking add (constraint product\_productid\_fk3 foreign key(product\_id) references product(product\_id));

alter table booking add constraint payment\_paymentid\_fk foreign key (payment\_id) references payment (payment\_id);

commit;



**Figure 3. 51 Booking**

SQL> select \* from booking;

Graphical user interface, text

Description automatically generated

**Figure 3. 52 Booking query.**

**Department table**

SQL> create table department (department\_id number(6), department\_name varchar(50), constraint departmentid\_pk primary key(department\_id));

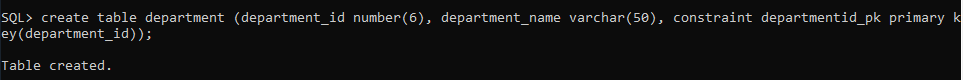
SQL> insert into department values (2, 'Airline Booking Department');

SQL> insert into department values (3, 'Cruise Booking Department');

SQL> insert into department values (4, 'Car Rental Department');

SQL> insert into department values (5, 'Insurance Booking Department');

SQL> insert into department values (1, 'Hotel Booking Department');



**Figure 3. 53 Department**

A screenshot of a computer

Description automatically generated with medium confidence

**Figure 3. 54 Insert statement.**

SQL> select \* from department;

Text

Description automatically generated

**Figure 3. 55 Department query**

**Employee table**

SQL> create table employee (employee\_id number(6), first\_name varchar2(50), last\_name varchar2(50), department\_id number(3), address varchar2(256), city varchar2(50), postal\_code varchar2(10), phone\_number varchar2(16), email varchar2(50));

SQL> insert into employee select employee\_id, first\_name, last\_name, department\_id, address, city, postal\_code, phone\_number, email from employee\_load;

SQL> alter table employee add (constraint employeeid\_pk primary key(employee\_id), constraint department\_departmentid\_fk foreign key(department\_id) references department(department\_id));

A screenshot of a computer

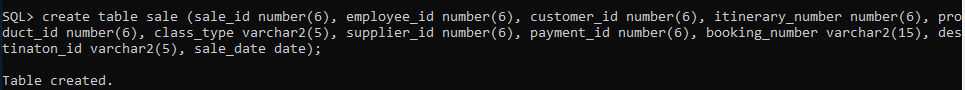
Description automatically generated with low confidence

**Figure 3. 56 Employee**

**Sale table**

SQL> create table sale (sale\_id number(6), employee\_id number(6), customer\_id number(6), itinerary\_number number(6), product\_id number(6), class\_type varchar2(5), supplier\_id number(6), payment\_id number(6), booking\_number varchar2(15), destinaton\_id varchar2(5), sale\_date date);

SQL> insert into sale select sale\_id, employee\_id, customer\_id, itinerary\_number, product\_id, class, supplier\_id, payment\_id, booking\_number, destination\_id, sale\_date from all\_sale\_load;



**Figure 3. 57 Sale**

SQL> alter table sale add constraint saleid\_pk primary key(sale\_id);

SQL> alter table sale add constraint employee\_employeeid\_fk foreign key(employee\_id) references employee(employee\_id);

SQL> alter table sale add constraint customer\_customerid\_fk foreign key(customer\_id) references customer(customer\_id);

SQL> alter table sale add constraint class\_class\_type\_fk foreign key(class\_type) references class(class\_type);

SQL> alter table sale rename column destinaton\_id to destination\_id;

SQL> alter table sale add constraint destination\_destinationid\_fk foreign key(destination\_id) references destination(destination\_id);

alter table sale add constraint payment\_paymentid\_fk1 foreign key (payment\_id) references payment(payment\_id);

delete from sale where supplier\_id = 150;

delete from sale where supplier\_id = 12;

select sale\_id from sale where supplier\_id=150;

select sale\_id from sale where supplier\_id=12;

delete from tax\_quote where sale\_id=60;

alter table sale add constraint MAIN\_SUPPLIER\_SUPPLIERID\_FK1 foreign key(supplier\_id) references main\_supplier (supplier\_id);

alter table sale add constraint booking\_booking\_number\_fk foreign key(booking\_number) references booking\_number(booking\_number);

alter table sale add constraint product\_productid\_fk1 foreign key (product\_id) references product (product\_id);

alter table sale add constraint itinerarynumber\_itinerary\_number\_fk1 foreign key(itinerary\_number) references itinerary\_number(itinerary\_number);

SQL> select sale\_id, employee\_id, customer\_id, booking\_number, itinerary\_number from sale order by sale\_id;

Text

Description automatically generated

**Figure 3. 58 Sale query**

**Tax quote table**

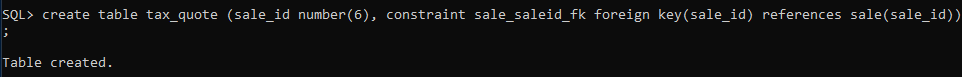
Tax quoted to agents by the supplier for different products.

SQL> create table tax\_quote (sale\_id number(6), constraint sale\_saleid\_fk foreign key(sale\_id) references sale(sale\_id));drop

SQL> alter table tax\_quote add constraint tax\_quoteid\_pk primary key(tax\_quote\_id);

SQL> alter table tax\_quote add tax\_quote\_id integer generated by default on null as identity;

Commit;



**Figure 3. 59 Tax quote**

SQL> select \* from tax\_quote;

A picture containing diagram

Description automatically generated

**Figure 3. 60 Tax quote query**

# **Workshop 5**

## **Update/progress of this project to TEA management**

Dear Travel Experts Management,

I have provided an update on the database creation project my team is currently working on. See below;

**i.** **Development and Evolution of Requirements**: To date, we have been gathering requirements and have made significant progress in defining the scope and objectives of the project. Our team has been working closely with stakeholders to ensure that we have a clear understanding of their needs and expectations. Based on this information, we have been able to refine and prioritize our requirements and are now moving forward with the development phase.

**ii.** **Timeline and Task Completion:** Our project timeline is on track, and we are making good progress in completing our tasks and milestones. Our team is working diligently to ensure that all tasks are completed on time and to a high standard. Despite some challenges and obstacles, we are confident that we will be able to meet the deadlines set out in our project plan.

**iii.** **Future Holds:** As we move forward, we will be focusing on the implementation of the project and testing our deliverables. Our team will also be working to ensure that we are meeting our stakeholders' needs and making any necessary adjustments to the project as we go. In the future, we anticipate that the project will result in a high-quality, user-friendly platform that will greatly improve the travel experience for our clients.

**iv.** **Areas of Concern:** Our team does have some concerns regarding the potential for technical difficulties and delays in the project. However, we have contingency plans in place to mitigate these risks and are confident that we will be able to overcome any challenges that arise.

Thank you for your continued support and please do not hesitate to contact me if you have any questions or concerns.

Sincerely,

Femi Adams

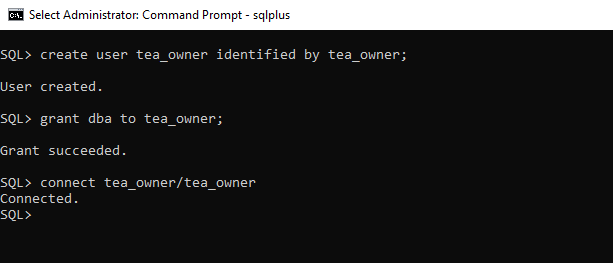
## **User Creation**

**Dba user creation**

SQL> create user tea\_owner identified by tea\_owner;

SQL> grant dba to tea\_owner;

SQL> connect tea\_owner/tea\_owner



**Figure 4. 1 Tea owner**

SQL> show user

Text

Description automatically generated

**Figure 4. 2 Show user**

**Senior agent role**

Script to show roles and privileges (select role, table\_name, privilege from role\_tab\_privs;)

Senior agents have privileges on all tables. He can view and insert on employee table. One senior agent (commission specialist) has all privilege on commission table

SQL> create role senior\_agent;

SQL> grant connect to senior\_agent;

SQL> grant create session to senior\_agent;

SQL> grant all on booking to senior\_agent;

SQL> grant all on class to senior\_agent;

SQL> grant all on commission to senior\_agent;

SQL> grant all on company to senior\_agent;

SQL> grant all on customer to senior\_agent;

SQL> grant all on cust\_credit\_card to senior\_agent;

SQL> grant all on department to senior\_agent;

SQL> grant all on destination to senior\_agent;

SQL> grant select, insert on employee to senior\_agent;

SQL> grant all on itinerary to senior\_agent;

SQL> grant all on main\_supplier to senior\_agent;

SQL> grant all on payment to senior\_agent;

SQL> grant all on payment\_type to senior\_agent;

SQL> grant all on product to senior\_agent;

SQL> grant all on sale to senior\_agent;

SQL> grant all on supplier to senior\_agent;

SQL> grant all on tax\_quote to senior\_agent;

SQL> select role, table\_name, privilege from role\_tab\_privs where role = 'SENIOR\_AGENT';

Text

Description automatically generated

**Figure 4. 3 Senior agent role**

Text

Description automatically generated

**Figure 4. 4 Senior agent query**

**Commission specialist**

Has same authorization with senior agents but has all access on commission table

SQL> create role commission\_specialist;

SQL> grant senior\_agent to commission\_specialist;

SQL> grant all on commission to commission\_specialist;

Text

Description automatically generated

Text

Description automatically generated

**Figure 4. 5 Commission specialist**

Text

Description automatically generated

**Figure 4. 6 Commission specialist query**

**Intermediate agent role**

Intermediate agent has all access to most tables. He can only view, employee, commission, department.

SQL> create role intermediate\_agent;

SQL> grant connect to intermediate\_agent;

SQL> grant create session to intermediate\_agent;

SQL> grant all on booking to intermediate\_agent;

SQL> grant all on class to intermediate\_agent;

SQL> grant select, insert on commission to intermediate\_agent;

SQL> grant all on cust\_credit\_card to intermediate\_agent;

SQL> grant select on department to intermediate\_agent;

SQL> grant all on destination to intermediate\_agent;

SQL> grant select on employee to intermediate\_agent;

SQL> grant all on itinerary to intermediate\_agent;

SQL> grant all on main\_supplier to intermediate\_agent;

SQL> grant all on payment to intermediate\_agent;

SQL> grant all on payment\_type to intermediate\_agent;

SQL> grant all on product to intermediate\_agent;

SQL> grant all on sale to intermediate\_agent;

SQL> grant all on supplier to intermediate\_agent;

SQL> grant select, insert on tax\_quote to intermediate\_agent;

SQL> select role, table\_name, privilege from role\_tab\_privs where role = 'INTERMEDIATE\_AGENT';

Text

Description automatically generated

**Figure 4. 7 Intermediate agent**

Graphical user interface, text

Description automatically generated

**Figure 4. 8 Intermediate agent query**

**Junior agent role**

Junior Agents can only view and insert on some tables and don’t have access to most tables.

SQL> create role junior\_agent;

SQL> grant create session to junior\_agent;

SQL> grant select, insert on booking to junior\_agent;

SQL> grant select, insert on class to junior\_agent;

SQL> grant select on company to junior\_agent;

SQL> grant select, insert on cust\_credit\_card to junior\_agent;

SQL> grant select, insert on itinerary to junior\_agent;

SQL> grant select on main\_supplier to junior\_agent;

SQL> grant select, insert on payment to junior\_agent;

SQL> grant select on product to junior\_agent;

SQL> grant select, insert on sale to junior\_agent;

SQL> grant select on supplier to junior\_agent;

SQL> grant select on tax\_quote to junior\_agent;

SQL> select role, table\_name, privilege from role\_tab\_privs where role = 'INTERMEDIATE\_AGENT';

Text

Description automatically generated

**Figure 4. 9 Junior agent**

Graphical user interface, text

Description automatically generated

**Figure 4. 10 Junior agent query**

**TEA manager role**

Has access to all tables, can create tables, can create views, can back up tables, can create users, can flashback any table, lock any table etc.

SQL> create role manager;

SQL> grant connect, resource to manager;

SQL> grant create user to manager;

SQL> grant alter user to manager;

SQL> grant create view to manager;

SQL> grant create table to manager;

SQL> grant create any table to manager;

SQL> grant drop any table to manager;

SQL> grant insert any table to manager;

SQL> grant flashback any table to manager;

SQL> grant alter any table to manager;

SQL> grant lock any table to manager;

SQL> grant backup any table to manager;

SQL> grant create any directory to manager;

SQL> grant drop any directory to manager;

SQL> grant create profile to manager;

SQL> grant alter profile to manager;

SQL> grant drop profile to manager;

SQL> grant create role to manager;

SQL> grant alter any role to manager;

SQL> grant create any table to manager;

SQL> grant alter any table to manager;

SQL> grant drop any table to manager;

SQL> grant insert any table to manager;

SQL> grant select any table to manager;

SQL> grant update any table to manager;

SQL> grant delete any table to manager;

SQL> grant all on booking to manager;

SQL> grant all on class to manager;

SQL> grant all on commission to manager;

SQL> grant all on company to manager;

SQL> grant all on customer to manager;

SQL> grant all on cust\_credit\_card to manager;

SQL> grant all on department to manager;

SQL> grant all on destination to manager;

SQL> grant all on employee to manager;

SQL> grant all on itinerary to manager;

SQL> grant all on main\_supplier to manager;

SQL> grant all on payment to manager;

SQL> grant all on payment\_type to manager;

SQL> grant all on product to manager;

SQL> grant all on sale to manager;

SQL> grant all on supplier to manager;

SQL> grant all on tax\_quote to manager;

Text

Description automatically generated

**Figure 4. 11 Manager role**

Text

Description automatically generated

**Figure 4. 12 Manager query**

**Owner role**

Has access to everything in the database

SQL> create role owner;

SQL> grant dba to owner;

Text

Description automatically generated with medium confidence

**Figure 4. 13 Grant statement**

SQL> SELECT \* FROM session\_privs order by privilege;

Text

Description automatically generated

**Figure 4. 14 Privilege query**

## **User creation and granted roles**

**Users**

SQL> create user aleshia identified by aleshia;

SQL> create user evan identified by evan;

SQL> create user france identified by france;

SQL> create user ulysses identified by ulysses;

SQL> create user Tyisha identified by Tyisha;

SQL> create user eric identified by eric;

SQL> create user marg identified by marg;

SQL> create user laquita identified by laquita;

SQL> create user laura identified by laura;

SQL> create user yuette identified by yuette;

SQL> create user fernanda identified by fernanda;

SQL> create user charlesetta identified by charlesetta;

SQL> create user corrinne identified by corrinne;

SQL> create user niesha identified by niesha;

SQL> create user charisse identified by charisse;

SQL> create user reuben identified by reuben;

SQL> create user michell identified by michell;

SQL> create user edgar identified by edgar;

Text

Description automatically generated

**Figure 4. 15 Create TEA users.**

SQL> select username, account\_status, default\_tablespace from dba\_users;

Text

Description automatically generated

**Figure 4. 16 Dba users query**

## **Grant role to users**

**Junior agent to users**

SQL> grant junior\_agent to aleshia;

SQL> grant junior\_agent to evan;

SQL> grant junior\_agent to france;

SQL> grant junior\_agent to ulysses;

SQL> grant junior\_agent to tyisha;

SQL> grant junior\_agent to eric;

Text

Description automatically generated

**Figure 4. 17 Junior agent grant**

**Intermediate agent to users**

SQL> grant intermediate\_agent to marg;

SQL> grant intermediate\_agent to laquita;

SQL> grant intermediate\_agent to laura;

SQL> grant intermediate\_agent to yuette;

SQL> grant intermediate\_agent to fernanda;

SQL> grant intermediate\_agent to charlesetta;

Text

Description automatically generated

**Figure 4. 18 Intermediate agent grant**

**Senior agent to users**

SQL> grant senior\_agent to corrinne;

SQL> grant senior\_agent to niesha;

SQL> grant senior\_agent to charisse;

Text

Description automatically generated

**Figure 4. 19 Senior agent grant**

**Commission Specialist to user**

SQL> grant commission\_specialist to reuben;

Text

Description automatically generated

**Figure 4. 20 Commission specialist grant**

**Manager to user**

SQL> grant manager to michell;

Text

Description automatically generated with low confidence

**Figure 4. 21 Manager grant**

**Owner to user**

SQL> grant owner to edgar;

Graphical user interface, text

Description automatically generated with medium confidence

**Figure 4. 22 Owner grant**

## **Profile creation**

SQL> create profile junior\_time\_limit limit connect\_time unlimited sessions\_per\_user 6 idle\_time 15 password\_lock\_time 3 failed\_login\_attempts 3;

SQL> create profile intermediate\_time\_limit limit connect\_time unlimited sessions\_per\_user 6 idle\_time 15 password\_lock\_time 3 failed\_login\_attempts 3;

SQL> create profile senior\_time\_limit limit connect\_time unlimited sessions\_per\_user 4 idle\_time 15 password\_lock\_time 3 failed\_login\_attempts 3;

SQL> create profile manager\_time\_limit limit connect\_time unlimited sessions\_per\_user 1 idle\_time unlimited password\_lock\_time unlimited failed\_login\_attempts unlimited;

SQL> create profile owner\_time\_limit limit connect\_time unlimited sessions\_per\_user 1 idle\_time unlimited password\_lock\_time unlimited failed\_login\_attempts unlimited;

A picture containing text

Description automatically generated

**Figure 4. 23 Profiles for existing roles**

SQL> show parameter resource\_limit;

A picture containing graphical user interface

Description automatically generated

**Figure 4. 24 Resource limit**

### Assign profile to users.

**Junior profile**

Junior time limit is assigned to junior agents

SQL> alter user aleshia profile junior\_time\_limit;

SQL> alter user evan profile junior\_time\_limit;

SQL> alter user france profile junior\_time\_limit;

SQL> alter user ulysses profile junior\_time\_limit;

SQL> alter user tyisha profile junior\_time\_limit;

SQL> alter user eric profile junior\_time\_limit;

Text

Description automatically generated

**Figure 4. 25 Junior time profile to existing users**

**Intermediate profile**

Profile assigned to intermediate agents

SQL> alter user marg profile intermediate\_time\_limit;

SQL> alter user laquita profile intermediate\_time\_limit;

SQL> alter user laura profile intermediate\_time\_limit;

SQL> alter user yuette profile intermediate\_time\_limit;

SQL> alter user fernanda profile intermediate\_time\_limit;

SQL> alter user charlesetta profile intermediate\_time\_limit;

Text

Description automatically generated

**Figure 4. 26 Intermediate time profile to existing users**

**Senior profile**

Profile assigned to senior agents.

SQL> alter user corrinne profile senior\_time\_limit;

SQL> alter user niesha profile senior\_time\_limit;

SQL> alter user charisse profile senior\_time\_limit;

SQL> alter user reuben profile senior\_time\_limit;

Text

Description automatically generated

**Figure 4. 27 Senior agent profile to existing users**

**Manager profile**

Profile assigned to senior manager.

SQL> alter user michell profile manager\_time\_limit;

Text

Description automatically generated

**Figure 4. 28 Manager time profile to existing user**

**Owner profile**

Profile assigned to senior owner

SQL> alter user edgar profile owner\_time\_limit;

Text

Description automatically generated

**Figure 4. 29 Owner time to existing user**

SELECT \* FROM DBA\_PROFILES WHERE PROFILE='&PROFILE\_NAME';

Text

Description automatically generated

**Figure 4. 30 Dba profile query**

# **Workshop 6**

## **Configuring listener.ora file**

**Directory**

Listener.ora file can be found in $ORACLE HOME\NETWORK\ADMIN directory C:\Oracle\db\19c\network\admin\listener.ora on the server.

**Steps to configuring Listener.ora file.**

* Add your hostname or dns name of the server the database is on. In this case I used dns name, but you can check for hostname by typing >hostname in cmd line.
* Add the port number the listener is listening on. I used the default port number 1521.
* Save the file and open cmd prompt.
* Use the listener command utility LSNRCTL to get into listener.
* Start the listener using START key word.
* Check the status of listener using STATUS keyword.
* Ensure that the listener is running and configured properly on the server.

Graphical user interface, text, application, email

Description automatically generated

**Figure 5. 1 Listener.ora file**

Text

Description automatically generated

**Figure 5. 2 Lsnrctl utility**

## **Configuring Tnsnames.ora File**

**Directory**

Tnsnames.ora file can be found in $ORACLE HOME\NETWORK\ADMIN directory C:\Oracle\db\19c\network\admin\tnsnames.ora

**Steps to configuring tnsnames.ora.**

* Ensure that the listener is running on the server side.
* Then on the client side, make sure that the Oracle client software is installed and running.
* Create a tnsnames.ora file in the clients$ORACLE HOME\NETWORK\ADMIN directory.
* Input the server’s hostname/dns name and port number and save the file.

Graphical user interface, text, application, email

Description automatically generated

**Figure 5. 3 Tnsnames.ora file**

## **Configuring sqlnet.ora file**

**Directory**

Sqlnet.ora file can be found in $ORACLE HOME\NETWORK\ADMIN directory C:\Oracle\db\19c\network\admin\sqlnet.ora

**Steps to configuring sqlnet.ora.**

* Create a sqlnet.ora file in $ORACLE HOME\NETWORK\ADMIN directory.
* Input the information in the screenshot below.
* Open cmd prompt and set sid to service name of the server database.
* Enter username and password as system/Password1@tea.

NB:(could not perform this test because the dns in class is down)

* You can also use a tnsping utility to test the connection.

Text, letter

Description automatically generated

**Figure 5. 4 Sqlnet.ora**

# **Workshop 7**

## **Data Warehouse & Data Mining**

### Data Warehouse Model/Scripts for Summary Views:

The data warehouse for the travel agency can be designed using a star schema, with fact tables such as Booking, Customer, Payment, and Destination, and dimension tables such as Time, Travel Package, and Traveler Information. The schema can be implemented using SQL scripts and can be optimized for reporting and analysis purposes.

### Metadata:

Metadata can be established to provide information about the data warehouse schema, tables, columns, and relationships. It can also include information about data sources, data transformations, and business rules applied during the ETL process. Metadata can be documented using a metadata repository or a data dictionary.

### Sample Reports:

Some sample reports that can be generated from the data warehouse to support marketing decision making are:

* Revenue analysis by destination, travel package, and time.
* Customer segmentation based on demographics and travel behavior.
* Sales performance by agent, region, and product.
* Top-selling products and destinations.
* Market share analysis by competitor and region.

### Marketing-Related Questions:

Data mining activities can help answer several marketing-related questions such as:

Which destinations and travel packages are popular among customers?

What factors influence customer buying behavior?

What are the trends in customer preferences and behaviors over time?

How do the agency's marketing and sales efforts affect customer acquisition and retention?

How can the agency improve customer experience and loyalty?

### Recommendations for Future Data Collection:

To enhance the data warehouse and support better decision-making, the travel agency should consider collecting additional data such as:

* Customer feedback and ratings on travel packages and destinations
* Social media data to understand customer sentiment and engagement.
* Web analytics data to analyze customer behavior on the agency's website.
* Competitive intelligence data to monitor industry trends and benchmark against competitors.
* Customer service data to measure satisfaction levels and identify areas for improvement.

# **Workshop 8**

## **Travel agency backup plan**

### Frequency of backups:

* During normal business hours (9:00 am to 9:00 pm), incremental backups should be taken every 2 hours.
* During busy times (sales, pre-Christmas, post-Christmas, summer, fall, thanksgiving) incremental backups should be taken every hour.
* During slow times, incremental backups should be taken every 4 hours.
* A full backup should be taken weekly.

### Recommended backup method:

* The recommended backup method is online backup using RMAN.
* The backup should include all tablespaces, including the additional tablespaces created for specific purposes.
* Backup sets should be split across multiple disks to minimize the risk of data loss.

### Recommendation concerning Archive/No archive mode with reasons:

* Archive log mode is recommended to ensure that all transactions are recoverable.
* The archive destination should be located on a separate disk from the database files to minimize the risk of data loss.
* Regular backups of the archive logs should be taken to ensure that the archive destination is also backed up.

### Recommendations concerning the number of redo log groups/members with reasons:

* Two groups with two members each are recommended to ensure that there is always a current redo log available.
* The size of the redo logs should be increased to accommodate the average number of transactions per redo log.

### Strategy for effective use of a tape drive (or separate device/location):

* The tape drive should be used to take a full back-up once a week.
* The tapes should be rotated offsite on a regular basis to minimize the risk of data loss in the event of a disaster.

### Exact backup schedule:

* Incremental backups every 2 hours during regular business hours.
* Incremental backups every hour during busy times.
* Incremental backups every 4 hours during slow times.
* Full back up once a week.
* Regular backups of the archive logs.
* The full backup should be taken to tape and rotated offsite on a regular basis.

## **Testing backup plan (Cold Backup & RMAN)**

### Cold backup

Simulated instance where user’s tablespace is deleted.

Text

Description automatically generated

**Figure 6. 1 Cold backup**

### RMAN Backup

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

**Figure 6. 2 Rman backup**

# **Workshop 9**

## **Sample invoice/itinerary script**

**Using joins**

SQL> select c.customer\_id, c.first\_name, c.last\_name, c.address, c.postal\_code, i.itinerary\_number, i.trip\_start, i.trip\_end, i.number\_of\_travellers, d.destination\_id, ms.supplier\_name, b.bill\_description, b.booking\_number, b.billed\_amount, b.total\_price, cc.card\_number from sale s

join customer c on s.customer\_id=c.customer\_id

join itinerary i on s.itinerary\_number=i.itinerary\_number

join destination d on s.destination\_id=d.destination\_id

join main\_supplier ms on s.supplier\_id=ms.supplier\_id

join booking b on s.booking\_number=b.booking\_number

join payment p on s.payment\_id=p.payment\_id

join cust\_credit\_card cc on p.cust\_credit\_card\_id=cc.cust\_credit\_card\_id;

Graphical user interface

Description automatically generated

**Figure 7. 1 Sample invoice**

**Stored procedure**

CREATE OR REPLACE PROCEDURE sp\_select\_customer (p\_cust\_details OUT SYS\_REFCURSOR)

IS

BEGIN

OPEN p\_cust\_details for SELECT c.customer\_id, c.first\_name, c.last\_name, c.address, c.postal\_code,

i.itinerary\_number, i.trip\_start, i.trip\_end, i.number\_of\_travellers,

d.destination\_id, ms.supplier\_name, b.bill\_description, b.booking\_number,

b.billed\_amount, b.total\_price, cc.card\_number

FROM sale s

JOIN customer c ON s.customer\_id=c.customer\_id

JOIN itinerary i ON s.itinerary\_number=i.itinerary\_number

JOIN destination d ON s.destination\_id=d.destination\_id

JOIN main\_supplier ms ON s.supplier\_id=ms.supplier\_id

JOIN booking b ON s.booking\_number=b.booking\_number

JOIN payment p ON s.payment\_id=p.payment\_id

JOIN cust\_credit\_card cc ON p.cust\_credit\_card\_id=cc.cust\_credit\_card\_id;

END;

/

CREATE OR REPLACE PROCEDURE get\_sale\_details

IS

-- Define cursor for the query

CURSOR sale\_cursor IS

SELECT c.customer\_id, c.first\_name, c.last\_name, c.address, c.postal\_code, i.itinerary\_number, i.trip\_start, i.trip\_end, i.number\_of\_travellers, d.destination\_id, ms.supplier\_name, b.bill\_description, b.booking\_number, b.billed\_amount, b.total\_price, cc.card\_number

FROM sale s

JOIN customer c ON s.customer\_id=c.customer\_id

JOIN itinerary i ON s.itinerary\_number=i.itinerary\_number

JOIN destination d ON s.destination\_id=d.destination\_id

JOIN main\_supplier ms ON s.supplier\_id=ms.supplier\_id

JOIN booking b ON s.booking\_number=b.booking\_number

JOIN payment p ON s.payment\_id=p.payment\_id

JOIN cust\_credit\_card cc ON p.cust\_credit\_card\_id=cc.cust\_credit\_card\_id;

-- Declare variables for the cursor

v\_customer\_id sale.customer\_id%TYPE;

v\_first\_name customer.first\_name%TYPE;

v\_last\_name customer.last\_name%TYPE;

v\_address customer.address%TYPE;

v\_postal\_code customer.postal\_code%TYPE;

v\_itinerary\_number itinerary.itinerary\_number%TYPE;

v\_trip\_start itinerary.trip\_start%TYPE;

v\_trip\_end itinerary.trip\_end%TYPE;

v\_number\_of\_travellers itinerary.number\_of\_travellers%TYPE;

v\_destination\_id destination.destination\_id%TYPE;

v\_supplier\_name main\_supplier.supplier\_name%TYPE;

v\_bill\_description booking.bill\_description%TYPE;

v\_booking\_number booking.booking\_number%TYPE;

v\_billed\_amount booking.billed\_amount%TYPE;

v\_total\_price booking.total\_price%TYPE;

v\_card\_number cust\_credit\_card.card\_number%TYPE;

BEGIN

-- Open the cursor

OPEN sale\_cursor;

-- Loop through the cursor and output the results

LOOP

FETCH sale\_cursor INTO v\_customer\_id, v\_first\_name, v\_last\_name, v\_address, v\_postal\_code, v\_itinerary\_number, v\_trip\_start, v\_trip\_end, v\_number\_of\_travellers, v\_destination\_id, v\_supplier\_name, v\_bill\_description, v\_booking\_number, v\_billed\_amount, v\_total\_price, v\_card\_number;

EXIT WHEN sale\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v\_customer\_id || ', ' || v\_first\_name || ' ' || v\_last\_name || ', ' || v\_address || ', ' || v\_postal\_code || ', ' || v\_itinerary\_number || ', ' || v\_trip\_start || ', ' || v\_trip\_end || ', ' || v\_number\_of\_travellers || ', ' || v\_destination\_id || ', ' || v\_supplier\_name || ', ' || v\_bill\_description || ', ' || v\_booking\_number || ', ' || v\_billed\_amount || ', ' || v\_total\_price || ', ' || v\_card\_number);

END LOOP;

-- Close the cursor

CLOSE sale\_cursor;

END;

/

Text

Description automatically generated

**Figure 7. 2 Stored procedure.**

SQL> SET SERVEROUTPUT ON;

SQL> EXEC get\_sale\_details;

Text

Description automatically generated

**Figure 7. 3 Stored procedure query.**

## **Commission report script**

select \* from commission;

A picture containing graphical user interface

Description automatically generated

**Figure 7. 4 Commission query**

## **Entry of a new customer**

SQL> CREATE SEQUENCE seq\_customers START WITH 403 INCREMENT BY 1 NOCACHE NOCYCLE;

ACCEPT first\_name PROMPT "Enter first name: "

ACCEPT last\_name PROMPT 'Enter last name: '

ACCEPT email PROMPT 'Enter email: '

ACCEPT home\_phone PROMPT 'Enter phone number: '

ACCEPT business\_phone PROMPT 'Enter business phone: '

ACCEPT date\_of\_birth PROMPT 'Enter DOB (Example 24-DEC-1998: '

ACCEPT address PROMPT 'Enter address: '

ACCEPT city PROMPT 'Enter city: '

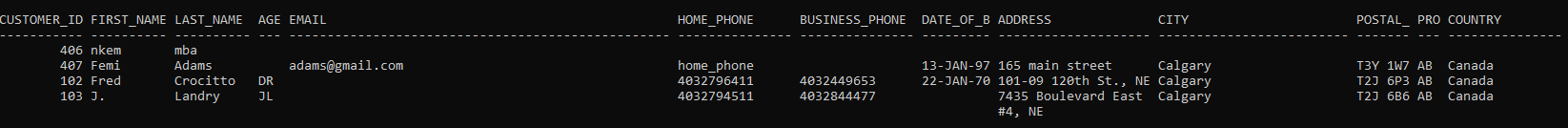
ACCEPT postal\_code PROMPT 'Enter postal code: '

ACCEPT province PROMPT 'Enter province: '

ACCEPT country PROMPT 'Enter country: '

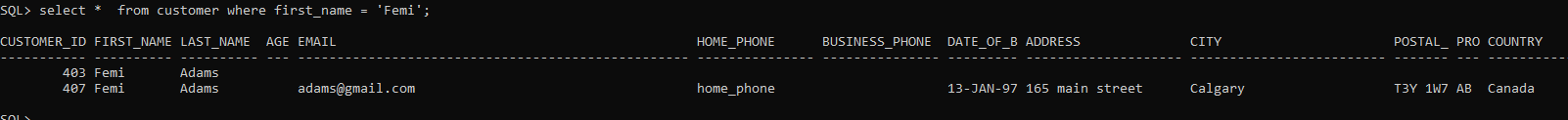
INSERT INTO customer (customer\_id, first\_name, last\_name, email, home\_phone, business\_phone, date\_of\_birth, address, city, postal\_code, province, country) VALUES (seq\_customers.NEXTVAL, '&first\_name', '&last\_name', '&email', 'home\_phone', '&business\_phone', '&date\_of\_birth', '&address', '&city', '&postal\_code', '&province', '&country');

@C:\Temp\addcustomer.sql



**Figure 7. 5 New customer inserted**

select \* from customer where first\_name = 'Femi';



**Figure 7. 6 New customer query**

# **Workshop – 10**

## **Performance & Tuning**

1. **Use indexing**: Indexing can significantly improve the performance of a database. Make sure that all the tables are indexed appropriately based on the type of queries you will be running.
2. **Optimize queries**: Poorly written queries can negatively impact the performance of a database. Make sure that all the queries are optimized and use the appropriate indexes.
3. **Use caching**: Caching can help reduce the load on the database by storing frequently accessed data in memory. Consider using a caching layer such as Redis or Memcached.
4. **Normalize the database**: Normalization is the process of organizing the data in a database to minimize redundancy. This can help improve the performance of the database and reduce the risk of data inconsistency.
5. **Use appropriate data types**: Make sure that you use the appropriate data types for the columns in your tables. Using the wrong data type can negatively impact the performance of the database.
6. **Set appropriate buffer sizes**: Make sure that you set appropriate buffer sizes for the database server. This can help improve the performance of the database.
7. **Use connection pooling**: Connection pooling can help reduce the overhead of establishing new database connections. Consider using a connection pooling library such as HikariCP or Apache Commons DBCP.
8. **Monitor database performance**: Regularly monitor the performance of the database to identify any potential bottlenecks or issues. Use monitoring tools such as Nagios or Zabbix to help with this.

# **Workshop 11**

# **Cover letter**

May 3, 2023

Travel Expert Agency,

123 Main Street T1Y 5R9,

Calgary,

Alberta.

Dear Management,

I am writing to submit a proposal for the development of a database and reporting system for Travel Experts Travel Agency. As a leading provider of software development services, we are confident that we can deliver a high-quality solution that meets the unique needs of your organization.

This proposal outlines our approach to the project, including the project objectives, scope, constraints, assumptions, approach, deliverables, project schedule, and budget. We have also included recommendations for next steps and progress reporting.

We appreciate your consideration of our proposal and look forward to the opportunity to work with your organization. Please do not hesitate to contact us if you have any questions or require additional information.

Sincerely,

Femi Adams

Dabasys Inc

# **Executive summary**

Travel Experts Travel Agency needs a database and reporting system that will allow them to store customer and supplier information, track sales and commissions, and identify marketing opportunities. They have allocated a budget of $500,000 for the development of the system, with the potential for other Canadian agencies to purchase it in the future.

Our proposal outlines our approach to the project, including the development of a database prototype using Oracle and the creation of a user-friendly graphical interface. We will work closely with the agency to understand their specific requirements and ensure that the system is designed to accommodate future enhancements and web integration.

Our team has extensive experience in software development and database design, and we are confident that we can deliver a high-quality solution that meets the needs of Travel Experts Travel Agency. We have outlined a project schedule and budget that will allow us to complete the project on time and within budget.

# **Background**

Travel Experts Travel Agency is a Calgary-based travel agency that employs 16 travel agents, 1 manager, and an owner. The agency has identified the need for a database and reporting system that will allow them to store customer and supplier information, track sales and commissions, and identify marketing opportunities. Although there are few travel industry-related software packages available, none of them meet the specific needs of the agency.

# **Business objectives**

The business objectives of Travel Experts Travel Agency are to:

1. Store customer and supplier information in a central database
2. Track sales and commissions to ensure proper calculations and identify outstanding commissions
3. Identify marketing opportunities using sales and customer data
4. Develop a user-friendly graphical interface for easy use by travel agents

# **Project objectives**

The project objectives of this proposal are to:

1. Develop a database prototype using Oracle that meets the specific needs of Travel Experts Travel Agency
2. Create a user-friendly graphical interface for easy data entry and reporting
3. Ensure that the system is designed to accommodate future enhancements and web integration
4. Work closely with the agency to ensure that their specific requirements are met
5. Complete the project on time and within budget

# **Scope**

The scope of the project includes the development of a database prototype using Oracle and the creation of a user-friendly graphical interface. The database will store customer and supplier information, track sales and commissions, and allow for the identification of marketing opportunities. The graphical interface will be designed to allow for easy data entry and reporting. The system will be designed to accommodate future enhancements.

Our team has evaluated various database systems and operating systems for the Travel Agency project, and after careful consideration, we recommend Oracle Database and Ubuntu Linux as the best software/hardware solution for the agency. In this proposal, we will outline the reasons for our recommendation and provide a detailed plan for the complete project.

### Database System:

We recommend using Oracle Database for the Travel Agency project. Oracle is a well-established database system that offers robust features for managing large amounts of data, ensuring data integrity and security, and providing scalability for future growth. It also provides advanced analytics and reporting capabilities that can be beneficial for the travel agency to gain insights into their business operations.

### Hardware Solution:

To ensure optimal performance of the Travel Agency database system, we recommend using the following hardware solution:

Processor: Intel Core i7 or higher

RAM: 16 GB or higher

Storage: Solid State Drive (SSD) with at least 500 GB capacity

Network Interface Card (NIC): Gigabit Ethernet

Backup Solution: RAID 5 array or higher

### Reasons for Selection:

We have selected Oracle Database and Ubuntu Linux based on the following factors:

1. Reliability: Oracle Database is known for its reliability, scalability, and data integrity. It also has a proven track record of being used by many businesses worldwide, including travel agencies.
2. Security: Oracle Database provides advanced security features such as encryption, user authentication, and access control, which are essential for protecting sensitive data such as customer information.
3. Performance: Oracle Database is optimized for high-performance and can handle large volumes of data with ease. Ubuntu Linux is known for its stability and ability to run resource-intensive applications such as databases.
4. Support: Both Oracle and Ubuntu have large communities of developers and users, and there is ample support available for any technical issues that may arise during the project.

## **Constraints**

There are some constraints that need to be considered during the development of the database prototype. These constraints include:

Budget: The project budget is $500,000 which means that the development team needs to work within this budget.

Timeframe: The project needs to be completed within a specific timeframe, and any delays may impact the overall success of the project.

Resources: The development team needs to have access to the necessary resources, including hardware, software, and personnel.

# **Assumptions**

To ensure the success of the project, the following assumptions have been made:

The development team has the necessary expertise to design and develop the database prototype using Oracle.

The travel agency will provide all the necessary information and data required for the database prototype.

The travel agency will provide feedback and guidance throughout the development process.

# **Approach**

The development team will follow a structured approach to design and develop the database prototype. The approach will include the following steps:

Requirements gathering: The team will gather all the requirements from the travel agency and document them.

**Design:** The team will design the database prototype based on the requirements gathered.

Development: The team will develop the database prototype using Oracle.

Testing: The team will test the database prototype to ensure that it meets the requirements and is free from errors.

Deployment: The team will deploy the database prototype to the travel agency.

User training: The team will provide user training to the travel agency staff to ensure that they can use the system effectively.

# **Deliverables**

The following deliverables will be provided as part of the project:

Requirements document: This document will outline all the requirements for the database prototype.

Design document: This document will provide the database prototype design.

Database prototype: The database prototype will be developed using Oracle.

Testing report: This report will provide the results of the testing conducted on the database prototype.

User manual: This manual will provide instructions on how to use the system.

User training: The development team will provide user training to the travel agency staff.

# **Project Schedule**

The project schedule is as follows:

1. Requirements gathering: 4 weeks
2. Design: 10 weeks
3. Development: 10 weeks
4. Testing: 3 weeks
5. Deployment: 1 week
6. User training: 2 weeks
7. Total project duration: 30 weeks

# **Budget**

The project budget is $500,000. The budget will be used to cover the costs associated with hardware, software, personnel, and other expenses.

### Hardware expenses

Server: $10,000

Network equipment (switches, routers, cables, etc.): $5,000

Desktop computers for travel agents and managers: $20,000

Printers and scanners: $5,000

Backup and storage devices: $10,000

Total hardware expenses: $50,000

### Software expenses:

Oracle Database Software: $20,000

Oracle Database Administrator Tool: $5,000

Development software (IDE, code repository, etc.): $5,000

Reporting software: $5,000

Anti-virus and security software: $5,000

Total software expenses: $40,000

### Personnel expenses:

Project Manager: $100,000

Database Developer: $80,000

Database Administrator: $80,000

Reporting Developer: $60,000

Help Desk Support: $60,000

Training Specialist: $50,000

Total personnel expenses: $430,000

### Other expenses:

Office space rent: $10,000

Office supplies and equipment: $5,000

Training materials and documentation: $5,000

Contingency (unforeseen expenses): $10,000

Total other expenses: $30,000

# **Recommendations**

Based on the requirements and constraints of the project, the development team recommends that the travel agency consider the following:

Providing access to all the necessary data and information required for the development of the database prototype.

Providing feedback and guidance throughout the development process to ensure that the final product meets their needs.

Allocating sufficient resources, including personnel, to ensure the success of the project.

# **Next Steps/ Progress Reporting**

The development team will provide progress reports to the travel agency on a regular basis throughout the project. The progress reports will provide updates on the status of the project, including any issues or risks that need to be addressed. The development team will also communicate any delays or changes to the project schedule or budget as soon as they are identified.

# **Appendix**

The appendix will include any additional information or documentation related to the project, such as technical specifications, diagrams, or charts.

# **Workshop 13**

## **Oracle Hot Backup Shell Script and a datapump backup script**

### Datapump backup script (oradp.sh)

#!/bin/bash

#

# oradp.sh - script to show datapump running

#

#

if [ $# != 1 ]; then

echo "Syntax Error. This code requires one parameter"

exit 1

fi

YYMMDD=`date +'%Y%m%d%H%M'`

ORACLE\_SID=$1

DP\_DIR=/datapump/${ORACLE\_SID}

DUMPFILE=dp\_${ORACLE\_SID}\_${YYMMDD}.dmp

LOGFILE=dp\_${ORACLE\_SID}\_${YYMMDD}.log

dbhome > /dev/null 2>&1

if [ $? != 0 ]; then

echo "Error: ${ORACLE\_SID} is not on this server"

exit 1

fi

ORAENV\_ASK=NO

. oraenv > /dev/null 2>&1

if [ $? != 0 ]; then

echo "Erro: Unable to set the environment"

exit 1

fi

#

#is the database open?

#

sqlplus -s /nolog << EOF > TEMP.txt

whenever sqlerror exit 1

connect / as sysdba

select status from v\$instance;

EOF

if [ $? != 0 ]; then

echo "Error: Unable to verify if the database is open"

exit 1

fi

if [ "`grep OPEN TEMP.txt`" != "OPEN" ]; then

echo "Error: Database is not OPEN"

exit 1

else

echo "Database verifed as being OPEN"

fi

#

#setup the location for datapumps ( /datapump/${ORACLE\_SID} )

#

if [ ! -d /datapump/${ORACLE\_SID} ]; then

mkdir ${DP\_DIR} > /dev/null 2>&1

if [ $? != 0 ]; then

echo "Error: unable to create /datapump/${ORACLE\_SID} for datapump backups"

exit 1

fi

fi

sqlplus -s /nolog << EOF > TEMP.txt

whenever sqlerror exit 1

connect system/Password1@${ORACLE\_SID}

set head off

create or replace directory DPEXP as '${DP\_DIR}';

EOF

if [ $? != 0 ]; then

echo "Error: setup the directory within the database"

exit 1

fi

expdp system/Password1 directory=DPEXP dumpfile=${DUMPFILE} logfile=${LOGFILE} full=y

if [ $? != 0 ]; then

echo "Errors reported in datapump"

exit 1

fi

echo "Compressing....... ${DUMPFILE}"

gzip ${DUMPFILE}

**Functions.sh script**

#

# These are rays functions

#

echo\_msg()

{

echo "$1" | tee -a ${LOGFILE}

}

### Hot backup shell script (orahot.sh)

#!/bin/bash

#

# orahot.sh - performs a backup (hot) of the database one tablespace at a time - good old fashioned backup

#

# this will not work with ASM systems and should be run on an instance level

#

# We have decided that hot backups backups will go to /hot\_backup

#

# Femi Adams - 27th April, 2023

#

. /home/oracle/scripts/functions.sh

if [ $# != 1 ]; then

echo "Syntax Error. This code requires one parameter"

exit 1

fi

YYMMDD=`date +'%Y%m%d%H%M'`

ORACLE\_SID=$1

BU\_LOCATION=/hot\_backup

BU\_FILES\_DIR=${BU\_LOCATION}/orahot\_${ORACLE\_SID}\_${YYMMDD}

export BU\_RECOVERY\_FILES=${BU\_FILES\_DIR}/orahot\_${ORACLE\_SID}\_recovery.sh

LOGFILE=/home/oracle/logs/orahot\_${ORACLE\_SID}\_${YYMMDD}.log

echo\_msg "LOGFILE used : ${LOGFILE}"

echo\_msg "orahot backup for $ORACLE\_SID started at `date`"

dbhome > /dev/null 2>&1

if [ $? != 0 ]; then

echo\_msg "Error: ${ORACLE\_SID} is not a database instance on this server"

exit 1

fi

ORAENV\_ASK=NO

. oraenv > /dev/null 2>&1

if [ $? != 0 ]; then

echo\_msg "Error: Unable to set the environmment"

exit 1

fi

#

# is the database open?

#

sqlplus -s /nolog << EOF > TEMP.txt

whenever sqlerror exit 1

connect / as sysdba

set head off

select status from v\$instance;

EOF

if [ $? != 0 ]; then

echo\_msg "Error: Unable to verify if the database is open"

exit 1

fi

if [ "`grep OPEN TEMP.txt`" != "OPEN" ]; then

echo\_msg "Error: Database is not OPEN"

exit 1

else

echo\_msg "Database verified as being OPEN"

fi

#

# verify the backup location

#

if [ -d ${BU\_LOCATION} ]; then

echo\_msg "Confirmed BU\_LOCATION ( ${BU\_LOCATION} ) exists"

else

echo\_msg "Confirmed BU\_LOCATION ( ${BU\_LOCATION} ) does not exists. Cancelling the backup"

exit 1

fi

mkdir ${BU\_FILES\_DIR} > /dev/null 2>&1

if [ $? != 0 ]; then

echo\_msg "Failed to create a good location for the hot backup - ${BU\_FILES\_DIR}"

exit 1

else

echo\_msg "Create a good location for the hot backup - ${BU\_FILES\_DIR}"

fi

#

# Ensure database is running and is in ARCHIVELOG mode

#

sqlplus -s /nolog << EOF > TEMP.txt

WHENEVER SQLERROR EXIT 1

connect / as sysdba

set head off

select log\_mode from v\$database where log\_mode='ARCHIVELOG';

EOF

if [ $? != 0 ]; then

echo "Error: Database not in Archive Mode"

exit 1

fi

if [ "`grep ARCHIVELOG TEMP.txt`" != "ARCHIVELOG" ]; then

echo "Error: Database is not in Archive Mode"

exit 1

else

echo "Database is in Archive Mode"

fi

#

# capture the current online log ---

#

sqlplus -s /nolog << EOF > TEMP.txt

WHENEVER SQLERROR EXIT 1

connect / as sysdba

set head off

select sequence# from v\$log where status='CURRENT';

EOF

START\_SEQ=`grep -v '^$' TEMP.txt`

echo "Current Sequence Number is ${START\_SEQ}"

#

# is there another backup running already?

#

sqlplus -s /nolog << EOF > TEMP.txt

WHENEVER SQLERROR EXIT 1

connect / as sysdba

set head off

select \* from v\$backup where status='ACTIVE';

EOF

if [ $? != 0 ]; then

echo "Another backup is running"

exit 1

else

echo "No other backup is running"

fi

#

# cycle through the data files backing them up one at a time to $BU\_FILES\_DIR

#

echo\_msg "Starting backup of datafiles....."

sqlplus -s /nolog << EOF > TEMP.txt

connect / as sysdba

whenever sqlerror exit 1

set pagesize 0

set linesize 2048

set heading off

set feedback off

set trimspool on

select tablespace\_name ||' '||file\_name||' '||' ${BU\_FILES\_DIR}/' || substr(file\_name, instr(file\_name,'/',-1,1)+1, length(file\_name) ) || '.' || file\_id

from dba\_data\_files

order by 1;

EOF

if [ $? != 0 ]; then

echo\_msg "Error querying database."

exit

fi

echo "# Files: " > ${BU\_FILES\_DIR}/TEMP.txt

grep -v '^$' TEMP.txt | while read TBL DTF BUF;

do

sqlplus -s /nolog << EOF > /dev/null

whenever sqlerror exit 1

connect / as sysdba

alter tablespace ${TBL} begin backup;

alter system switch logfile;

EOF

echo "copying ${DTF} to ${BUF}"

cp "${DTF}" "${BUF}"

echo "copying ${DTF} ${BUF}" >> ${BU\_FILES\_DIR}/TEMP.txt

echo "#recovering ${BUF} ${DTF}" >> ${BU\_RECOVERY\_FILES}

echo\_msg "Recovery file created: ${BU\_RECOVERY\_FILES}"

sqlplus -s /nolog << EOF > /dev/null

whenever sqlerror exit 1

connect / as sysdba

alter tablespace ${TBL} end backup;

EOF

if [ $? != 0 ]; then

echo\_msg "Datafiles not backed up."

exit 1

else

echo\_msg "Datafiles backup completed."

fi

done

#

# force log switch

#

sqlplus -s /nolog << EOF > /dev/null 2>&1

whenever sqlerror exit 1

connect / as sysdba

alter system switch logfile;

EOF

if [ $? != 0 ]; then

echo\_msg "Error: Unable to force log switch"

exit 1

else

echo\_msg "Log switch forced"

fi

#

# capture the current online log ---

#

sqlplus -s /nolog << EOF > TEMP.txt

WHENEVER SQLERROR EXIT 1

connect / as sysdba

set head off

select sequence# from v\$log where status='CURRENT';

EOF

END\_SEQ=`grep -v '^$' TEMP.txt`

echo "New Sequnce Number is ${END\_SEQ}"

#

# identify and copy all the archives to the $BU\_FILES\_DIR

#

echo\_msg "Starting backup of archivelogs....."

sqlplus -s /nolog << EOF > TEMP.txt

connect / as sysdba

whenever sqlerror exit 1

set pagesize 0

set linesize 2048

set heading off

set feedback off

set trimspool on

select name||';'||'${BU\_FILES\_DIR}/'||substr(name, instr(name,'/',-1,1)+1, length(name)) from v\$archived\_log where sequence# between ${START\_SEQ} and ${END\_SEQ};

EOF

if [ $? != 0 ]; then

echo\_msg "Archive log list not found."

exit

fi

while read -r line;

do

archive=$(echo "$line" | cut -d ";" -f1)

destination=$(echo "$line" | cut -d ";" -f2)

echo "copying ${archive} to ${destination}"

cp "$archive" "$destination"

echo "recovering ${destination} ${archive}" >> ${BU\_RECOVERY\_FILES}

echo\_msg "Recovery file created: ${BU\_RECOVERY\_FILES}"

if [ $? != 0 ]; then

echo\_msg "Archive logs not backed up."

exit 1

else

echo\_msg "Backup of archivelogs completed."

fi

done < TEMP.txt

#

# Back up the control file (both binary and logical)

#

sqlplus -s /nolog << EOF > TEMP.txt

connect / as sysdba

whenever sqlerror exit 1

alter database backup controlfile to trace as '$BU\_FILES\_DIR/controlfile.trc' reuse;

alter database backup controlfile to '$BU\_FILES\_DIR/controlfile.bin';

exit;

EOF

if [ $? != 0 ]; then

echo\_msg "Control file backup failed."

exit 1

else

echo "Control file backed up successfully to $BU\_FILES\_DIR"

fi

echo\_msg "orahot backup for $ORACLE\_SID ended successfully at `date`"

# **REFERENCES**

## **Workshop 3**

**Step 6:** [catalog.sql and catproc.sql — oracle-tech](https://community.oracle.com/tech/developers/discussion/1102366/catalog-sql-and-catproc-sql)

**Step 7:** [catalog.sql and catproc.sql — oracle-tech](https://community.oracle.com/tech/developers/discussion/1102366/catalog-sql-and-catproc-sql)

**Step 8:** [pupbld.sql — oracle-tech](https://community.oracle.com/tech/developers/discussion/2387813/pupbld-sql)

**Step 9:** [v$controlfile (oracle.com)](https://docs.oracle.com/en/database/oracle/oracle-database/19/refrn/V-CONTROLFILE.html#GUID-C285E8C5-6A2D-44B1-BDF8-F880B5F088C4)

**Step 10:** [v$datafile (oracle.com)](https://docs.oracle.com/en/database/oracle/oracle-database/19/refrn/V-DATAFILE.html#GUID-7BF7955C-9705-40F4-B2F6-5D7F3A32DD30)

**Step 11:** [v$logfile (oracle.com)](https://docs.oracle.com/en/database/oracle/oracle-database/19/refrn/V-LOGFILE.html#GUID-AE0B85B3-6B51-4FBD-A460-2D7090D9E0B5)