**Banking Transaction Management System (BTMS)**

**Database Designed Document**

**----------------------------------------------------------------**

**Version 08.25**

**08/05/2025**

**1. Introduction**

Banking Transaction Management System (BTMS) is a banking system that aims to bring satisfactory transaction processes, account management, and security enforcement within a banking environment for users who hold a bank account. An overview of the different processes needs to be translated into a database so bank authorities can take good decisions to respond to issues related to banking and improve the banking system. The overall purpose of such a database is to create a robust database that will assist end users in decision-making.

**2. Overview**

This report outlines the design and implementation of the Banking Transaction Management System (BTMS), a system intended to optimize transaction processing, ensure account security, and streamline account management in banking institutions. The design incorporates best practices in relational database design and Oracle technologies.

**3. Literature Review**

Research and projects in similar domains indicate the criticality of data consistency, security, and scalable architecture in banking systems. This project draws inspiration from established models in banking information systems and academic case studies on relational databases.

**4. Assumptions, Constraints, and Risks**

4.1 Assumptions

The system is assumed to operate within a virtual lab environment provided by UMGC, using Oracle 19c DBMS. It is assumed that users interacting with the system have basic database literacy.

4.2 Constraints

The system focuses solely on backend development using SQL. No GUI or frontend integrations are included. DCL and stored procedures are out of scope.

4.3 Risks

Potential risks include data redundancy if normalization is not enforced, loss of referential integrity if foreign key constraints are not maintained, and security threats if audit trails are not implemented adequately.

**5. Design Decisions**

5.1 Key Factors Influencing Design

Key factors include system performance, referential integrity, data consistency, scalability, and ease of maintenance. The use of surrogate keys and audit columns ensures tracking of modifications. Views and indexes improve read performance.

5.2 Functional Design Decisions

The system accepts insertions of customer, account, transaction, and branch data, with validation rules on account types, transaction types, and relationships. It produces reports via SQL queries and views. Data input is transactional, and queries support both analytics and auditing.

5.3 DBMS Decisions

Oracle 19c was selected for its enterprise-grade features and compatibility with the Virtual Lab Environment. Oracle SQL Developer served as the development IDE. Sequence and trigger mechanisms were used for surrogate key generation.

5.4 Security and Privacy Design Decisions

Surrogate key tracking and audit columns are employed for accountability. Views abstract sensitive audit data from end-user access. Only business-relevant attributes are exposed through views.

5.5 Performance and Maintenance Design Decisions

Indexes on frequently queried fields such as foreign keys, balance, transaction type support fast querying. Views support summary reporting. Data partitioning is logical, with foreign key constraints ensuring data consistency. Sequences auto-generate primary keys to minimize human error. Maintenance strategies include view abstraction and normalized design for scalability.

**6. Statement of Work**

6.1 Executive Summary

Banking Transaction Management System (BTMS) is a banking system that aims to bring satisfactory transaction processes, account management, and security enforcement within a banking environment for users who hold a bank account. An overview of the different processes needs to be translated into a database so bank authorities can make informed decisions to respond to issues related to banking and improve the banking system. The overall purpose of such a database is to create a robust system that will assist end users in decision-making.

6.2 Objectives

To satisfy the purpose of building a robust BTMS database, the project will be divided into four goals. And each goal will be split into measurable and achievable steps.

1. SOW – Statement of Work – Week 4

Develop a comprehensive document (minimum 1–2 pages) outlining the necessity of creating, designing, and implementing the BTMS database. The SOW should include:

* Executive Summary: A concise overview detailing the significance of the project and its intended impact.
* Purpose and Objectives: Clearly define the database’s role, its goals, and expected outcomes.
* Technical Specifications: Provide a thorough explanation of the technologies involved, including database design diagrams, DBMS selection, hardware and software requirements, and essential DDL and DML components.

1. Create a requirement definition document – Week 6

In this section of the project, the business requirements and business rules will be specified.

1. Describe each entity and its main attributes.
2. Describe each relationship and the cardinality from both directions of the relationship including business rules
3. Describe any other assumptions and special considerations you may have.

1. Design BTMS database by creating the Entity Relationship Diagram (ERD). - Week 6
2. Create an initial conceptual level ERD of BTMS which mainly captures and represents the description of business rules.

1. Refine the initial model and create design level ERD by normalization and resolving all many-to-many relationships.

* The ERD for the BMTS will consist of 5 entities where each entity should have at least 5 attributes. Draw appropriate relationships to connect related entities.
* The crow's feet notation is required.
* SQL Developer Data Modeler is the chosen diagramming tool.

1. Database implementation – Week 9 &11
2. DDL – First Part
3. Drop statements for all objects in the lab project
4. Add DDL Create statements for all tables and associated objects.
5. Now that your tables have been created, you can add indexes to certain columns to speed up queries.
6. DDL – Second Part
7. First modify your table structure to add some audit columns so that you can keep track of who adds/changes a record and when. Use DDL ALTER TABLE statement
8. Then read about SQL views and create a view for each table so that only business columns are included but not audit columns
9. use surrogate key instead of natural key as primary key.
10. create a row level trigger for each table so that you can populate surrogate key and audit columns with appropriate values.
11. Now check the DBMS data dictionary to make sure all your objects have been created successfully.
12. DML – Data Insertion
13. create SQL INSERT statements to populate each table with business data. Each table should have a minimum of 10 rows unless you have specific business rules that prevent it from having that many records. Make sure your sequences and triggers are valid and enabled so that surrogate keys and audit columns can be populated automatically.
14. Develop SQL SELECT statements to query your tables with a minimum of 20 SQL select statements. Query 1 to 12 are basic queries, plus at least 8 advanced queries. Each query should have a comment/description to explain its business purpose, as well as which requirement item you are satisfied with.

6.3 Project Scope

* In-scope
* Design an ERD - Entity-Relationship Diagram of the BMTS database
* Specify business rules and requirements
* Create conceptual, logical and physical ERD
* Implement Trigger to populate the surrogate key and audit columns
* Query BMTS database tables to retrieve values
* Out-of-scope
* SAVEPOINT is the only Transaction Control Language (TCL) command that is excluded
* Data Control Language commands such as grand or revoke are not needed
* Stored procedure will not be implemented in this project.
* No backend and front-end implementations. Focus is only BMTS database creation

6.4 Database Goals, Expectations and Deliverables

The database that will be created will be robust BMTS using cutting-edge technologies. The end users of the product will expect a database system with five entities, but it is designed to be extensible with new entities, etc.

* Technical report that includes SOW and ERD of the BTMS
* SQL file of the DDL related to the BTMS
* SQL file of the DML related to the BTMS

6.5 Database Benefits

The database we aim to build should enhance operational efficiency and ensure secure handling of financial transactions. This system minimizes fraud risks and provides a scalable architecture for future growth. In addition, the end users will use the BMTS database to produce information to help them in decision-making.

6.6 Project Hardware and Software Tools

* Project Hardware

To create BTMS, the Virtual Lab Environment will be used as a platform to conduct this database project. It is a hardware or operating system that is already built by the University of Maryland Global Campus (UMGC). Specifically, the host machine of UMGC is Microsoft Windows 11 Pro and version is 10.0.2200 Build 220000.

* Software Tools
* Diagramming Tool Identified
* ER-Assistant Version 2.10 was used for the creation of the ERD Diagram
* Cow’s feet annotation is the style used to for cardinality and connectivity
* Database and Access Method Identified
* Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
* Oracle Database 19c was the platform for implementing DDL that resulted in Data Dictionary and DML for data insertion into tables and querying
* Oracle IDE: Oracle SQL Developer - Version21.2.1.204.1703
* Oracle SQL Developer was the tool for implementation of DDL and DML
* Client will start the engine of the UMGC virtual lab that will create a connection. Then the client will connect to the UMGC virtual lab. The product, which is a standalone application, is located on the desktop.
* Office Productivity Tools: Microsoft Office 365, running on Windows

6.7 SQL Usage and Style Guide

* Drop scripts were implemented for tables, then sequences before creating tables and sequences
* “ON DELETE CASCADE” were added to the child tables
* Along with entity and referential integrities, domain constraints have been for all tables, specifically, “Check” that limits set of values.
* SQL Optimization: Proper indexing for possible natural key, foreign key columns and others and query optimization for performance.
* Creation of couple Views for security, customization and data consistency and reusability
* Primary Key: surrogate keys were generated instead of using natural keys
* Triggers were defined for each primary key, specifically “BEFORE INSERT”
* Minimum of ten (10) scripts were created for each table for data insertion
* Naming Conventions: Clear, descriptive table and column names following industry best practices along with space after each block of scripts.

6.8 Student-Defined Sections and Additional Content

* Mockaroo tool for data generation during data insertion into tables
* Data conversion from Data to Timestamp
* AI assistance for in-depth knowledge of some concepts such as M:N relationships, views, triggers, subqueries, etc.

**7. Requirements Analysis**

This section outlines the entities, relationships, and business rules defined for the Banking Transaction Management System (BTMS). Each entity contains at least five attributes. Cardinality and key constraints are also specified, ensuring database normalization and referential integrity.

**EMPLOYEE**

* Entity Description: employees who works at a branch location
* Primary Key:
* Attribute Name: EmployeeID
* Attribute Description: employee id
* Foreign Key:
* Attribute Name: BranchID
* Attribute Description: branch id
* Main attributes of EMPLOYEE:
* Attribute Name: F\_Name
* Attribute Description: first name.
* Attribute Name: L\_Name
* Attribute Description: last name.
* Attribute Name: Email
* Attribute Description: email
* Attribute Name: Position
* Attribute Description: position
* Attribute Name: H\_Date
* Attribute Description: Hire Date
* Relationships:
* “works” between EMPLOYEE and BRANCH
* Cardinality:
* 1:M between BRANCH and EMPLOYEE
* Business rules:
* A branch can have many employees; an employee works at one and only one branch

**BRANCH**

* Entity description: branch location of the bank that provides services
* Primary Key:
* Attribute Name: BranchID
* Attribute Description: branch id
* Main attributes of BRANCH:
* Attribute Name: Br\_Name
* Attribute Description: branch name
* Attribute Name: location
* Attribute Description: location
* Attribute Name: Mn\_Name
* Attribute Description: manager name
* Attribute Name: P\_Number
* Attribute Description: phone number
* Attribute Name: Est\_Date
* Attribute Description: established date
* Relationship:
* “works” between EMPLOYEE and BRANCH
* “executes” between BRANCH and OPERATION
* Cardinality:
* 1:M between BRANCH and EMPLOYEE
* 1:M between BRANCH and OPERATION
* Business rule:
* A branch can have many employees; an employee works at one and only one branch
* A branch can execute many operations; an operation may be executed by at most one branch

**CUSTOMER**

* Entity description: Customer of the bank who holds an account
* Primary Key:
* Attribute Name: CustomerID
* Attribute Description: customer id
* Main attributes of CUSTOMER:
* Attribute Name: F\_Name
* Attribute Description: first name
* Attribute Name: L\_Name
* Attribute Description: last name
* Attribute Name: DOB
* Attribute Description: date of birth
* Attribute Name: Address
* Attribute Description: address
* Attribute Name: P\_Number
* Attribute Description: phone number
* Attribute Name: Email
* Attribute: email
* Relationship:
* “holds” between CUSTOMER and ACCOUNT
* Cardinality:
* 1:M between CUSTOMER and ACCOUNT
* Business rule:
* A customer can hold many accounts; an account is held by one and only one customer

**ACCOUNT**

* Entity Description: Bank account that is hold by a customer
* Primary Key:
* Attribute Name: AccountID
* Attribute Description: account id or account number
* Foreign Key
* CustomerID
* Attribute Description: customer id
* Main attributes of ACCOUNT:
* Attribute Name: Balance
* Attribute Description: balance
* Attribute Name: Acct\_Type
* Attribute Description: account type
* Attribute Name: Crt\_Date
* Attribute Description: creation date
* Attribute Name: Status
* Attribute Description: status: active or inactive.
* Relationship:
* “holds” between CUSTOMER and ACCOUNT
* “has” between ACCOUNT and TRANSACTION
* Cardinality:
* 1:M between CUSTOMER and ACCOUNT
* 1:M between ACCOUNT and TRANSACTION
* Business rule:
* A customer can hold many accounts; an account is held by one and only one customer
* An account can have zero to many transactions; a transaction is attached to one and only one account

**TRANSACTION**

* Entity Description: Transaction that is carried out by customers
* Primary Key:
* Attribute Name: TransactionID
* Attribute Description: transaction id
* Foreign Keys:
* Attribute Name: AccountID
* Attribute Description: account id or bank account number
* Main attributes of Description:
* Attribute Name: Amount
* Attribute Description: amount
* Attribute Name: Tr\_Type
* Attribute Description: transaction type
* Attribute Name: Timestamp
* Attribute Description: timestamp
* Attribute Name: Description
* Attribute Description: description
* Relationship:
* “has” between ACCOUNT and TRANSACTION
* “involves” between TRANSACTION and OPERATION
* Cardinality:
* 1:M between ACCOUNT and TRANSACTION
* 1:M between TRANSACTION and OPERATION
* Business rules:
* An account can have zero to many transactions; a transaction is attached to one and only one account
* A transaction can involve many operations; an operation is linked by one and only one transaction

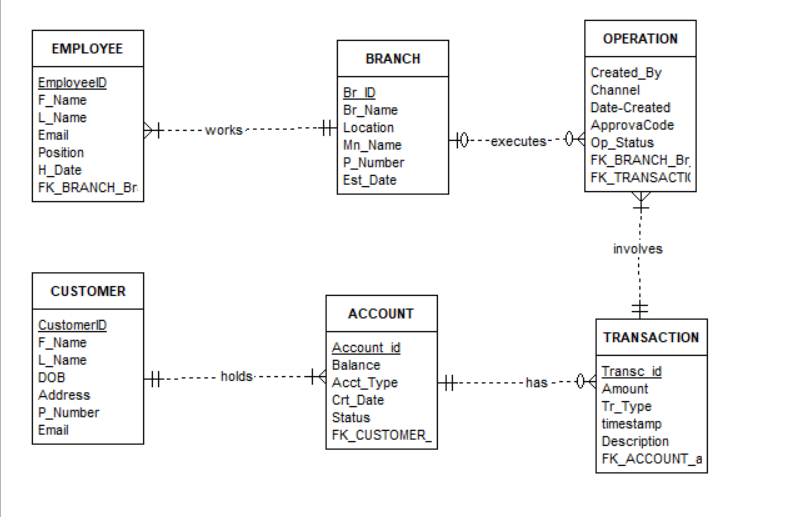
**OPERATION**

* Entity Description: Operation where a transaction can be carried out by a customer at a branch
* Primary Key:
* Attribute Name: BranchID & TransactionID
* Attribute Description: transaction id and transaction id
* Foreign Keys:
* Attribute Name: BranchID
* Attribute Description: branch
* Attribute Name: TransactionID
* Attribute Description: transaction id
* Main attributes of Description:
* Attribute Name: Channel
* Attribute Description: channel
* Attribute Name: Created\_by
* Attribute Description: created by
* Attribute Name: Date\_created
* Attribute Description: date created
* Attribute Name: Appvl\_Code
* Attribute Description: approval code
* Relationship:
* “executes” between BRANCH and OPERATION
* “involves” between TRANSACTION and OPERATION
* Cardinality:
* 1:M between BRANCH and OPERATION
* 1:M between TRANSACTION and OPERATION
* Business rules:
* A branch can execute many operations; an operation may be executed by at most one branch
* A transaction can involve many operations; an operation is linked by one and only one transaction

**8. Detailed Database Design**

8.1 Entity Relationship Diagram (ERD)

Refer to the ERD diagram below which visually represents the five main entities and their relationships using crow’s foot notation.



8.2 DDL Source Code

The following Data Definition Language (DDL) script defines the structure of the BTMS database:

set echo on;

--------------------------------------------------------------------------------

----------- 1. Dropping all tables ---------------------------------------------

DROP TABLE operation;

DROP TABLE employee;

DROP TABLE branch;

DROP TABLE transaction;

DROP TABLE account;

DROP TABLE customer;

-- Dropping all sequence

DROP SEQUENCE seq\_emp\_id;

DROP SEQUENCE seq\_br\_id;

DROP SEQUENCE seq\_transc\_id;

DROP SEQUENCE seq\_acct\_id;

DROP SEQUENCE seq\_cust\_id;

--------------------------------------------------------------------------------

------------- 2. Create tables for banking system ---------------------------------

CREATE TABLE customer

( cust\_id NUMBER(10) NOT NULL,

f\_name VARCHAR2(50) NOT NULL,

l\_name VARCHAR2(50) NOT NULL,

dob DATE NOT NULL,

address VARCHAR2(100) NOT NULL,

ph\_number CHAR(10) NOT NULL,

email VARCHAR2(50) NOT NULL,

CONSTRAINT customer\_pk PRIMARY KEY (cust\_id)

);

CREATE TABLE account

( account\_id NUMBER(10) NOT NULL,

balance DECIMAL(38,2) CHECK (balance >= 0) NOT NULL,

acct\_type VARCHAR2(50) CHECK (acct\_type IN ('Savings', 'Checking', 'Loan')) NOT NULL,

creat\_date DATE NOT NULL,

status VARCHAR2(50) NOT NULL,

fk\_customer\_cust\_id NUMBER(10),

CONSTRAINT account\_pk PRIMARY KEY (account\_id),

CONSTRAINT fk\_customer FOREIGN KEY (fk\_customer\_cust\_id)

REFERENCES customer(cust\_id) ON DELETE CASCADE

);

CREATE TABLE transaction

( transc\_id NUMBER(10) NOT NULL,

amount DECIMAL(38,2) NOT NULL,

transc\_type varchar2(50) CHECK (transc\_type IN ('Deposit', 'Withdrawal', 'Transfer')) NOT NULL,

timestamp TIMESTAMP NOT NULL,

description VARCHAR2(200) NOT NULL,

fk\_account\_account\_id NUMBER(10) NOT NULL,

CONSTRAINT transaction\_pk PRIMARY KEY (transc\_id),

CONSTRAINT fk\_account FOREIGN KEY (fk\_account\_account\_id) REFERENCES account(account\_id) ON DELETE CASCADE

);

CREATE TABLE branch

( br\_id NUMBER(10) NOT NULL,

br\_name VARCHAR2(50) NOT NULL,

location VARCHAR2(50) NOT NULL,

mgr\_name VARCHAR2(50) NOT NULL,

ph\_number CHAR(10) NOT NULL,

est\_date DATE,

CONSTRAINT branch\_pk PRIMARY KEY (br\_id)

);

CREATE TABLE operation

( date\_created TIMESTAMP NOT NULL,

created\_by varchar2(50) NOT NULL,

channel VARCHAR(50) NOT NULL,

appr\_code VARCHAR2(50) NULL,

op\_status VARCHAR2(50) NOT NULL,

fk\_branch\_br\_id NUMBER(10) NOT NULL,

fk\_transaction\_transc\_id NUMBER(10) NOT NULL,

CONSTRAINT operation\_pk PRIMARY KEY (fk\_branch\_br\_id, fk\_transaction\_transc\_id),

CONSTRAINT fk\_branch\_par FOREIGN KEY (fk\_branch\_br\_id) REFERENCES branch(br\_id) ON DELETE CASCADE,

CONSTRAINT fk\_transaction FOREIGN KEY (fk\_transaction\_transc\_id) REFERENCES transaction(transc\_id) ON DELETE CASCADE

);

CREATE TABLE employee

( emp\_id NUMBER(10) NOT NULL,

f\_name VARCHAR2(50) NOT NULL,

l\_name VARCHAR2(50) NOT NULL,

email VARCHAR2(50) NOT NULL,

position VARCHAR2(50) NOT NULL,

hired\_date DATE NOT NULL,

fk\_branch\_br\_id NUMBER(10),

CONSTRAINT employee\_pk PRIMARY KEY (emp\_id),

CONSTRAINT fk\_branch FOREIGN KEY (fk\_branch\_br\_id) REFERENCES branch(br\_id)

ON DELETE CASCADE

);

--------------------------------------------------------------------------------

----- 3. Create indexes for possible natural key, foreign key columns and others

-- Create unique index on natural key columns.

CREATE UNIQUE INDEX idx\_emp\_fname\_lname ON employee(f\_name, l\_name);

CREATE UNIQUE INDEX idx\_cust\_fname\_lname ON customer(f\_name, l\_name);

-- Create index on foreign key columns.

CREATE INDEX idx\_acct\_cust\_id ON account(fk\_customer\_cust\_id);

CREATE INDEX idx\_transc\_acct\_id ON transaction(fk\_account\_account\_id);

CREATE INDEX idx\_op\_transc\_id ON operation(fk\_transaction\_transc\_id);

CREATE INDEX idx\_op\_br\_id ON operation(fk\_branch\_br\_id);

CREATE INDEX idx\_emp\_br\_id ON employee(fk\_branch\_br\_id);

-- Create index on other columns that will be frequently used as query filters

CREATE INDEX idx\_balance ON account(balance);

CREATE INDEX idx\_transc\_type ON transaction(transc\_type);

CREATE INDEX idx\_op\_created\_by ON operation(created\_by);

---------------------------------------------------------------------------------

----- 4. Create views that are appropriate to support business requirements -----

/\*

Business purpose: This view simplifies access to all accounts that are currently active. It is useful for operational dashboards and reporting

\*/

-- View showing all accounts with 'Active' status

CREATE OR REPLACE VIEW view\_active\_acct AS

SELECT account\_id, fk\_customer\_cust\_id, acct\_type, balance, creat\_date

FROM account

Where status = 'Active';

/\*

Business purpose: This view provides a consolidated report of the customer details and the sum of all their account balances.

It is useful for customer and financial analysis

\*/

-- This view combines customer details with their account balances

CREATE OR REPLACE VIEW view\_cust\_acct\_summary AS

SELECT c.cust\_id, c.f\_name, c.l\_name, c.email, c.ph\_number, NVL(SUM(a.balance), 0) AS total\_balance

FROM customer c LEFT JOIN account a ON c.cust\_id = a.fk\_customer\_cust\_id

GROUP BY c.cust\_id, c.f\_name, c.l\_name, c.email, c.ph\_number;

/\*

Business purpose: This view provides crucial information about each branch. It will help allocate resources proportional

Also, it is useful to further understand the causes of frequentation to make better decision.

\*/

-- This view consist to know the frequentation of each branch by customers

CREATE OR REPLACE VIEW view\_br\_performance AS

SELECT t.location, t.br\_name, t.channel, COUNT(t.location) AS number\_of\_frequention

FROM (

SELECT b.location, b.br\_name, o.channel

FROM branch b INNER JOIN operation o ON b.br\_id = o.fk\_branch\_br\_id

WHERE o.channel IN ('Teller', 'ATM')) t

GROUP BY t.location, t.br\_name, t.channel;

------------------------------------------------------------------------------------

----- 5. Create sequences for the primary keys created in each entities ------------

-- Generate values for cust\_id (primary key of customer entity) through sequence

CREATE SEQUENCE seq\_cust\_id

MINVALUE 1

MAXVALUE 100

START WITH 1

INCREMENT BY 1

CACHE 20;

-- Generate values for br\_id (primary key of branch entity) through sequence

CREATE SEQUENCE seq\_br\_id

MINVALUE 101

MAXVALUE 200

START WITH 101

INCREMENT BY 1

CACHE 20;

-- Generate values for emp\_id (primary key of employee entity) through sequence

CREATE SEQUENCE seq\_emp\_id

MINVALUE 201

MAXVALUE 400

START WITH 201

INCREMENT BY 1

CACHE 20;

-- Generate values for account\_id (primary key of account entity) through sequence

CREATE SEQUENCE seq\_acct\_id

MINVALUE 401

MAXVALUE 600

START WITH 401

INCREMENT BY 1

CACHE 20;

-- Generate values for transc\_id (primary key transaction entity) through sequence

CREATE SEQUENCE seq\_transc\_id

MINVALUE 601

MAXVALUE 800

START WITH 601

INCREMENT BY 1

CACHE 20;

------------------------------------------------------------------------------------

----- 6. Create triggers ----------------------------------------------------------

CREATE OR REPLACE TRIGGER trigger\_cust\_id

BEFORE INSERT ON customer

FOR EACH ROW

BEGIN

IF :NEW.cust\_id IS NULL THEN

:NEW.cust\_id := seq\_cust\_id.NEXTVAL;

END IF;

END;

/

CREATE OR REPLACE TRIGGER trigger\_acct\_id

BEFORE INSERT ON account

FOR EACH ROW

BEGIN

IF :NEW.account\_id IS NULL THEN

:NEW.account\_id := seq\_acct\_id.NEXTVAL;

END IF;

END;

/

CREATE OR REPLACE TRIGGER trigger\_emp\_id

BEFORE INSERT ON employee

FOR EACH ROW

BEGIN

IF :NEW.emp\_id IS NULL THEN

:NEW.emp\_id := seq\_emp\_id.NEXTVAL;

END IF;

END;

/

CREATE OR REPLACE TRIGGER trigger\_br\_id

BEFORE INSERT ON branch

FOR EACH ROW

BEGIN

IF :NEW.br\_id IS NULL THEN

:NEW.br\_id := seq\_br\_id.NEXTVAL;

END IF;

END;

/

CREATE OR REPLACE TRIGGER trigger\_transc\_id

BEFORE INSERT ON transaction

FOR EACH ROW

BEGIN

IF :NEW.transc\_id IS NULL THEN

:NEW.transc\_id := seq\_transc\_id.NEXTVAL;

END IF;

END;

/

/\* The business value of this trigger is to ensure auditability for transaction records.

-- It automatically logs the staff who performed the transaction and when, helping meet compliance and tracking requirements.

\*/

CREATE OR REPLACE TRIGGER trg\_operation\_audit

BEFORE INSERT ON operation

FOR EACH ROW

BEGIN

:NEW.date\_created := CAST(SYSDATE AS TIMESTAMP);

END;

/

/\* The business value of this trigger is to ensure auditability of when the customer initiates the transaction.

It also ensures that the bank meet compliance and tracking requirements.

\*/

CREATE OR REPLACE TRIGGER trg\_transaction\_audit

BEFORE INSERT ON Transaction

FOR EACH ROW

BEGIN

:NEW.timestamp := CAST(SYSDATE AS TIMESTAMP);

END;

/

-- To confirm all your tables are present

SELECT TABLE\_NAME FROM USER\_TABLES;

-- To check the status and types of all your created objects (valid/invalid)

SELECT OBJECT\_NAME, STATUS, CREATED, LAST\_DDL\_TIME FROM USER\_OBJECTS;

8.3 DML and Query Source Code

The following Data Manipulation Language (DML) script populates the BTMS database with realistic data entries:

set echo on;

-- DML- SQL INSERT statement that populate CUSTOMER table with sample data. CUSTOMER data consists of 10 rows.

INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Alicia', 'Mills', TO\_DATE('1991-05-24', 'YYYY-MM-DD'), '19 Garden Ave, Austin', '5125551001', '[alicia.mills@example.com](mailto:alicia.mills@example.com)');

INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Brian', 'Tucker', TO\_DATE('1988-11-07', 'YYYY-MM-DD'), '82 Ridge Rd, Denver', '3035554521', '[brian.tucker@example.com](mailto:brian.tucker@example.com)');

INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Clara', 'Nguyen', TO\_DATE('1995-02-16', 'YYYY-MM-DD'), '34 Willow St, Seattle', '2065553122', '[clara.nguyen@example.com](mailto:clara.nguyen@example.com)');

INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Derek', 'Ford', TO\_DATE('1990-09-12', 'YYYY-MM-DD'), '106 Main Blvd, Miami', '7865559982', '[derek.ford@example.com](mailto:derek.ford@example.com)');

INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Ella', 'Brown', TO\_DATE('1986-03-04', 'YYYY-MM-DD'), '501 Pacific St, Portland', '5035557411', '[ella.brown@example.com](mailto:ella.brown@example.com)');

INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Frank', 'Lopez', TO\_DATE('1993-07-19', 'YYYY-MM-DD'), '212 Walnut Ln, Chicago', '7735552188', '[frank.lopez@example.com](mailto:frank.lopez@example.com)');

INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Grace', 'Kim', TO\_DATE('1989-12-02', 'YYYY-MM-DD'), '45 Sunset Ave, LA', '2135553011', '[grace.kim@example.com](mailto:grace.kim@example.com)');

INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Henry', 'Singh', TO\_DATE('1992-06-28', 'YYYY-MM-DD'), '87 Liberty St, Boston', '6175559003', '[henry.singh@example.com](mailto:henry.singh@example.com)');

INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Isla', 'Patel', TO\_DATE('1994-08-10', 'YYYY-MM-DD'), '15 Beacon Way, New York', '7185556020', '[isla.patel@example.com](mailto:isla.patel@example.com)');

INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Jack', 'Davis', TO\_DATE('1996-10-05', 'YYYY-MM-DD'), '68 Pine Circle, Dallas', '2145557777', '[jack.davis@example.com](mailto:jack.davis@example.com)');

-- DML- SQL INSERT statement that populate ACCOUNT table with sample data. ACCOUNT data consists of 25 rows.

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (2300.50, 'Savings', TO\_DATE('2022-01-10', 'YYYY-MM-DD'), 'Active', 1);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (150.75, 'Checking', TO\_DATE('2022-03-22', 'YYYY-MM-DD'), 'Active', 2);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (6700.00, 'Loan', TO\_DATE('2021-09-05', 'YYYY-MM-DD'), 'Closed', 3);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (980.20, 'Savings', TO\_DATE('2022-07-13', 'YYYY-MM-DD'), 'Active', 4);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (3120.00, 'Checking', TO\_DATE('2023-01-01', 'YYYY-MM-DD'), 'Active', 5);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (4900.00, 'Loan', TO\_DATE('2021-12-15', 'YYYY-MM-DD'), 'Delinquent', 6);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (1125.35, 'Savings', TO\_DATE('2022-04-22', 'YYYY-MM-DD'), 'Active', 7);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (305.60, 'Checking', TO\_DATE('2022-08-09', 'YYYY-MM-DD'), 'Frozen', 8);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (15000.00, 'Loan', TO\_DATE('2021-06-21', 'YYYY-MM-DD'), 'Active', 9);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (220.10, 'Savings', TO\_DATE('2023-03-11', 'YYYY-MM-DD'), 'Active', 10);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (705.90, 'Checking', TO\_DATE('2022-02-28', 'YYYY-MM-DD'), 'Active', 1);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (9900.00, 'Loan', TO\_DATE('2022-05-15', 'YYYY-MM-DD'), 'Active', 2);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (650.00, 'Savings', TO\_DATE('2023-06-20', 'YYYY-MM-DD'), 'Active', 3);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (120.00, 'Checking', TO\_DATE('2022-09-01', 'YYYY-MM-DD'), 'Frozen', 4);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (4200.00, 'Loan', TO\_DATE('2021-03-03', 'YYYY-MM-DD'), 'Closed', 5);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (1580.30, 'Savings', TO\_DATE('2023-04-12', 'YYYY-MM-DD'), 'Active', 2);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (875.40, 'Checking', TO\_DATE('2022-10-05', 'YYYY-MM-DD'), 'Active', 4);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (2400.00, 'Loan', TO\_DATE('2020-12-01', 'YYYY-MM-DD'), 'Active', 6);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (130.20, 'Savings', TO\_DATE('2023-07-01', 'YYYY-MM-DD'), 'Active', 8);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (395.90, 'Checking', TO\_DATE('2021-08-30', 'YYYY-MM-DD'), 'Closed', 9);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (7600.00, 'Loan', TO\_DATE('2023-05-23', 'YYYY-MM-DD'), 'Active', 10);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (999.99, 'Savings', TO\_DATE('2022-11-18', 'YYYY-MM-DD'), 'Active', 2);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (543.21, 'Checking', TO\_DATE('2023-02-14', 'YYYY-MM-DD'), 'Frozen', 3);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (8200.00, 'Loan', TO\_DATE('2021-01-01', 'YYYY-MM-DD'), 'Delinquent', 5);

INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (510.00, 'Savings', TO\_DATE('2023-08-08', 'YYYY-MM-DD'), 'Active', 1);

-- DML- SQL INSERT statement that populate TRANSACTION table with sample data. TRANSACTION data consists of 15 rows.

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (200.00, 'Deposit', 'Initial deposit', 401);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES(100.00, 'Withdrawal', 'ATM withdrawal', 402);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (300.00, 'Transfer', 'Transfer to savings', 403);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (50.00, 'Withdrawal', 'POS Purchase', 404);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (1000.00, 'Deposit', 'Salary', 405);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (25.00, 'Withdrawal', 'ATM fee', 403);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (500.00, 'Deposit', 'Freelance payment', 407);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (75.00, 'Transfer', 'Transfer to friend', 408);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (120.00, 'Deposit', 'Cash deposit', 403);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (200.00, 'Withdrawal', 'Bill payment', 410);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (500.00, 'Deposit', 'Initial deposit', 415);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (150.00, 'Withdrawal', 'ATM withdrawal', 423);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (300.00, 'Transfer', 'Transfer to savings', 420);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (75.00, 'Withdrawal', 'POS purchase', 423);

INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (600.00, 'Transfer', 'Loan repayment', 425);

-- DML- SQL INSERT statement that populate BRANCH table with sample data. BRANCH data consists of 10 rows.

INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Downtown', 'New York', 'Alice Miles', '2125551010', TO\_DATE('2010-01-01','YYYY-MM-DD'));

INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Midtown', 'Chicago', 'Brian Lee', '3125551020', TO\_DATE('2012-05-15','YYYY-MM-DD'));

INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Uptown', 'Houston', 'Carla Yu', '7135551030', TO\_DATE('2014-03-20','YYYY-MM-DD'));

INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Central', 'Los Angeles', 'David Singh', '2135551040', TO\_DATE('2011-08-10','YYYY-MM-DD'));

INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Westside', 'San Diego', 'Eva Knight', '6195551050', TO\_DATE('2013-09-01','YYYY-MM-DD'));

INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Southpoint', 'Miami', 'Fred Owen', '3055551060', TO\_DATE('2015-10-12','YYYY-MM-DD'));

INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Riverside', 'Denver', 'Grace Patel', '3035551070', TO\_DATE('2016-07-21','YYYY-MM-DD'));

INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Northend', 'Seattle', 'Henry Kim', '2065551080', TO\_DATE('2018-02-14','YYYY-MM-DD'));

INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Eastside', 'Atlanta', 'Isla Brown', '4045551090', TO\_DATE('2017-04-25','YYYY-MM-DD'));

INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Metro', 'Boston', 'Jack Chen', '6175551100', TO\_DATE('2019-12-31','YYYY-MM-DD'));

-- DML- SQL INSERT statement that populate EMPLOYEE table with sample data. EMPLOYEE data consists of 10 rows.

INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Angela', 'White', '[awhite@example.com](mailto:awhite@example.com)', 'Teller', TO\_DATE('2018-04-01','YYYY-MM-DD'), 101);

INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES('Ben', 'King', '[bking@example.com](mailto:bking@example.com)', 'Manager', TO\_DATE('2017-03-15','YYYY-MM-DD'), 102);

INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Chris', 'Davis', '[cdavis@example.com](mailto:cdavis@example.com)', 'Clerk', TO\_DATE('2019-06-22','YYYY-MM-DD'), 103);

INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Diana', 'Morris', '[dmorris@example.com](mailto:dmorris@example.com)', 'Security', TO\_DATE('2020-01-10','YYYY-MM-DD'), 104);

INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Ethan', 'Scott', '[escott@example.com](mailto:escott@example.com)', 'Teller', TO\_DATE('2016-08-12','YYYY-MM-DD'), 105);

INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Fiona', 'Hill', '[fhill@example.com](mailto:fhill@example.com)', 'Clerk', TO\_DATE('2015-02-27','YYYY-MM-DD'), 106);

INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('George', 'Young', '[gyoung@example.com](mailto:gyoung@example.com)', 'Manager', TO\_DATE('2014-11-03','YYYY-MM-DD'), 107);

INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Hannah', 'Green', '[hgreen@example.com](mailto:hgreen@example.com)', 'Clerk', TO\_DATE('2021-05-18','YYYY-MM-DD'), 108);

INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Ian', 'Lopez', '[ilopez@example.com](mailto:ilopez@example.com)', 'Teller', TO\_DATE('2013-09-09','YYYY-MM-DD'), 109);

INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Judy', 'Turner', '[jturner@example.com](mailto:jturner@example.com)', 'Manager', TO\_DATE('2012-12-12','YYYY-MM-DD'), 110);

-- DML- SQL INSERT statement that populate OPERATION table with sample data. OPERATION data consists of 20 rows.

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Angela White', 'Teller', 'AP123', 'Approved', 101, 601); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Ben King', 'ATM', 'AP124', 'Completed', 102, 602); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Chris Davis', 'Online', 'AP125', 'Approved', 103, 603); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Diana Morris', 'Teller', 'AP126', 'Rejected', 104, 604); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Ethan Scott', 'ATM', 'AP127', 'Approved', 105, 605); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Fiona Hill', 'Online', 'AP128', 'Approved', 106, 606); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('George Young', 'Teller', 'AP129', 'Completed', 107, 607); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Hannah Green', 'ATM', 'AP130', 'Approved', 108, 608); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Ian Lopez', 'Online', 'AP131', 'Approved', 109, 609); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Judy Turner', 'Teller', 'AP132', 'Pending', 110, 610); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Angela White', 'ATM', 'AP133', 'Approved', 101, 611); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Ben King', 'Online', 'AP134', 'Completed', 102, 612);

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Chris Davis', 'Teller', 'AP135', 'Approved', 103, 613); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Diana Morris', 'ATM', 'AP136', 'Completed', 104, 614);

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Ethan Scott', 'Online', 'AP137', 'Approved', 105, 615); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Fiona Hill', 'Teller', 'AP138', 'Rejected', 106, 601); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('George Young', 'ATM', 'AP139', 'Pending', 107, 602); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Hannah Green', 'Online', 'AP140', 'Completed', 108, 603); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Ian Lopez', 'Teller', 'AP141', 'Approved', 109, 604); --

INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Judy Turner', 'ATM', 'AP142', 'Approved', 110, 605); --

commit;

/\*

20 SQL Queries : 12 Basic, 8 Advanced

\*/

-- Query 1: select all columns and all rows from one table

-- Business purpose: this query selects all information about all customers

select \* from customer;

-- Query 2: Select five columns and all rows from one table

-- Business purpose: this query selects all information from five attributes about all branch

select br\_name as "Branch Name", location as "Location", mgr\_name as "Manager Name", ph\_number as "Phone Number", est\_date as "Establishment Date" from branch;

--Query 3: Select all columns from all rows from one view

-- Business purpose: this selects all information about a created view called view\_active\_acct

select \* from view\_active\_acct;

--Query 4: Using a join on 2 tables, select all columns and all rows from the tables without the use of a Cartesian product

-- Business purpose: this query joins all information from two table where there is a matching value on the customer id

select \* from customer c inner join account a on c.cust\_id = a.fk\_customer\_cust\_id;

--Query 5: Select and order data retrieved from one table

-- Business purpose: this query select all information of all employee in ascending order based on employee name

select \* from employee order by f\_name, l\_name asc;

--Query 6: Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 10 rows

-- Business purpose: this query joins information of attributes from customer, account and transaction limited to five attributes each and the first ten rows.

select c.f\_name as first\_name, c.l\_name as last\_name, c.address, c.ph\_number as phone\_number, c.email,

a.account\_id as account\_number, a.balance, a.acct\_type as account\_type, a.creat\_date as creation\_date, a.status,

t.transc\_id as transaction\_number, t.amount, t.transc\_type as transaction\_type, t.timestamp, t.description

from

customer c join account a on c.cust\_id = a.fk\_customer\_cust\_id join transaction t on a.account\_id = t.fk\_account\_account\_id

fetch first 10 rows only;

--Query 7: Select distinct rows using joins on 3 tables

-- Business purpose: select all distinct information from branches, employees, operation after joining them.

select distinct \*

from branch b join employee e on b.br\_id = e.fk\_branch\_br\_id join operation o on b.br\_id = o.fk\_branch\_br\_id;

--Query 8: Use GROUP BY and HAVING in a select statement using one or more tables

-- Business purpose: select all information from accounts that are grouped by account type where the totals by per account type is greater than 1000.00

select acct\_type as account\_type, sum(balance) as total\_balance from account group by acct\_type having sum(balance) > 1000.00;

--Query 9: Use IN clause to select data from one or more tables

-- Business purpose: select all information of three attributes after joining accounts and transactions.

select a.balance, a.acct\_type as account\_type, a.status, t.amount, t.transc\_type as transaction\_type, t.description

from account a join transaction t on a.account\_id = t.fk\_account\_account\_id

where transc\_type in ('Deposit', 'Withdrawal');

--Query 10: Select length of one column from one table (use LENGTH function)

-- Business purpose: this query calculates the length of the name of employees who execute transactions.

select length(created\_by) as name\_length from operation;

--Query 11: Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement.

-- Make sure you use ROLLBACK afterwards so that the data will not be physically removed

-- Business purpose: this query delete a record of information from customer data, then rollback the deleted record.

select \* from customer;

delete from customer where cust\_id = 1;

select \* from customer;

rollback;

--Query 12: Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement.

-- Make sure you use ROLLBACK afterwards so that the data will not be physically removed

-- Business purpose: this query changes the first name and last name of the record that has cutomer id set to 1 from customer data, then rollback to initial values.

select \* from customer;

update customer set f\_name = 'Donald', l\_name = 'Keita' where cust\_id = 2;

select \* from customer;

rollback;

--Perform 8 Additional Advanced Queries

-- Query 13: use GROUP BY count the number of branch in each location from view called "view\_br\_performance"

-- Business purpose: this query computes the number of branch by location from already created view, "view\_br\_performance"

select location, count(br\_name) as "BRANCHES PER LOCATION" from view\_br\_performance group by location ;

-- Query 14: use GROUP BY to show the number transaction per customer

-- Business purpose: this query compute the number of transaction carried out by customer

select c.f\_name as "First Name", c.l\_name as "Last Name", count(transc\_id) as "Number of Transaction"

from customer c join account a on c.cust\_id = a.fk\_customer\_cust\_id join transaction t on a.account\_id = t.fk\_account\_account\_id

group by c.f\_name, c.l\_name;

-- Query 15:show the five active account that were lastly created ORDER BY and FIRST n ROWS ONLY

-- Business purpose: this query select the last five created account that are still active

select account\_id as "Account number", creat\_date as "Creation Date", status as "Status"

from account where status = 'Active' order by creat\_date desc fetch first 5 rows only;

-- Query 16: Show the five first customers who two account by using JOIN and operator NOT IN.

-- Business purpose: this query show information about customer who have both Checking and Savings accounts

select c.f\_name || ' ' || c.l\_name as "Customer Name", account\_id as "Account number", a.acct\_type as "Account Type"

from customer c join account a on c.cust\_id = a.fk\_customer\_cust\_id

where a.acct\_type not in ('Loan');

-- Query 17: use JOIN and HAVING to aggregate the total amount of transaction per account.

-- Business purpose: this query the total amount of transaction per account with a total of more $500.

select a.account\_id, sum(t.amount)

from account a join transaction t on a.account\_id = t.fk\_account\_account\_id

group by a.account\_id having sum(t.amount) > 500;

-- Query 18: use LEFT JOIN to show all information all transaction and those done at branch

-- Business purpose: this script show transaction information along with the one that

select tp.fk\_account\_account\_id as "ACCOUNT NUMBER", tp.amount, tp.description, tp.channel

from

(select t.fk\_account\_account\_id, t.amount, t.description, o.channel

from transaction t left join operation o on t.transc\_id = o.fk\_transaction\_transc\_id

where o.channel in ('ATM', 'Teller')) tp;

-- Query 19: use SUBQUERY to show the name and location who carried out operations that have "Rejected" status

-- Business purpose:this query select employees that assisted customers resulting in rejected operations and the locations and branches where they work

select e.f\_name as "First Name", e.l\_name as "Last Name",

(select b.location

from branch b

where b.br\_id = e.fk\_branch\_br\_id) as "Branch Location"

from employee e

where e.f\_name || ' ' || e.l\_name in (

select o.created\_by

from operation o

where o.op\_status = 'Rejected'

);

-- Query 20: use JOIN Customer, Account, Transaction and Operation tables to show customers who carried out operation where

-- transaction\_type is "Withdrawal" and channel is "ATM"

-- Business purpose: this query select all customers that withdraw money from ATM machines

select c.f\_name || ' ' || c.l\_name as "Customer Name", t.transc\_type as "Transaction Type", o.channel as "Channel", t.amount as "Amount"

from customer c join account a on c.cust\_id = a.fk\_customer\_cust\_id join transaction t on a.account\_id = t.fk\_account\_account\_id

join operation o on t.transc\_id = o.fk\_transaction\_transc\_id

where t.transc\_type = 'Withdrawal' and o.channel = 'ATM';

8.4 DDL, DML, and Query Outputs

Below are sample outputs from executing the DDL and DML scripts:

-- DDL Output --

SQL>

SQL> --------------------------------------------------------------------------------

SQL> ----------- 1. Dropping all tables ---------------------------------------------

SQL> DROP TABLE operation;

Table OPERATION dropped.

SQL> DROP TABLE employee;

Table EMPLOYEE dropped.

SQL> DROP TABLE branch;

Table BRANCH dropped.

SQL> DROP TABLE transaction;

Table TRANSACTION dropped.

SQL> DROP TABLE account;

Table ACCOUNT dropped.

SQL> DROP TABLE customer;

Table CUSTOMER dropped.

SQL>

SQL> -- Dropping all sequence

SQL> DROP SEQUENCE seq\_emp\_id;

Sequence SEQ\_EMP\_ID dropped.

SQL> DROP SEQUENCE seq\_br\_id;

Sequence SEQ\_BR\_ID dropped.

SQL> DROP SEQUENCE seq\_transc\_id;

Sequence SEQ\_TRANSC\_ID dropped.

SQL> DROP SEQUENCE seq\_acct\_id;

Sequence SEQ\_ACCT\_ID dropped.

SQL> DROP SEQUENCE seq\_cust\_id;

Sequence SEQ\_CUST\_ID dropped.

SQL>

SQL> --------------------------------------------------------------------------------

SQL> ------------- 2. Create tables for banking system ---------------------------------

SQL> CREATE TABLE customer

2 ( cust\_id NUMBER(10) NOT NULL,

3 f\_name VARCHAR2(50) NOT NULL,

4 l\_name VARCHAR2(50) NOT NULL,

5 dob DATE NOT NULL,

6 address VARCHAR2(100) NOT NULL,

7 ph\_number CHAR(10) NOT NULL,

8 email VARCHAR2(50) NOT NULL,

9 CONSTRAINT customer\_pk PRIMARY KEY (cust\_id)

10 );

Table CUSTOMER created.

SQL>

SQL> CREATE TABLE account

2 ( account\_id NUMBER(10) NOT NULL,

3 balance DECIMAL(38,2) CHECK (balance >= 0) NOT NULL,

4 acct\_type VARCHAR2(50) CHECK (acct\_type IN ('Savings', 'Checking', 'Loan')) NOT NULL,

5 creat\_date DATE NOT NULL,

6 status VARCHAR2(50) NOT NULL,

7 fk\_customer\_cust\_id NUMBER(10),

8 CONSTRAINT account\_pk PRIMARY KEY (account\_id),

9 CONSTRAINT fk\_customer FOREIGN KEY (fk\_customer\_cust\_id)

10 REFERENCES customer(cust\_id)

11 ON DELETE CASCADE

12 );

Table ACCOUNT created.

SQL>

SQL> CREATE TABLE transaction

2 ( transc\_id NUMBER(10) NOT NULL,

3 amount DECIMAL(38,2) NOT NULL,

4 transc\_type varchar2(50) CHECK (transc\_type IN ('Deposit', 'Withdrawal', 'Transfer')) NOT NULL,

5 timestamp TIMESTAMP NOT NULL,

6 description VARCHAR2(200) NOT NULL,

7 fk\_account\_account\_id NUMBER(10) NOT NULL,

8 CONSTRAINT transaction\_pk PRIMARY KEY (transc\_id),

9 CONSTRAINT fk\_account FOREIGN KEY (fk\_account\_account\_id) REFERENCES account(account\_id)

10 ON DELETE CASCADE

11 );

Table TRANSACTION created.

SQL>

SQL> CREATE TABLE branch

2 ( br\_id NUMBER(10) NOT NULL,

3 br\_name VARCHAR2(50) NOT NULL,

4 location VARCHAR2(50) NOT NULL,

5 mgr\_name VARCHAR2(50) NOT NULL,

6 ph\_number CHAR(10) NOT NULL,

7 est\_date DATE,

8 CONSTRAINT branch\_pk PRIMARY KEY (br\_id)

9 );

Table BRANCH created.

SQL>

SQL> CREATE TABLE operation

2 ( date\_created TIMESTAMP NOT NULL,

3 created\_by varchar2(50) NOT NULL,

4 channel VARCHAR(50) NOT NULL,

5 appr\_code VARCHAR2(50) NULL,

6 op\_status VARCHAR2(50) NOT NULL,

7 fk\_branch\_br\_id NUMBER(10) NOT NULL,

8 fk\_transaction\_transc\_id NUMBER(10) NOT NULL,

9 CONSTRAINT operation\_pk PRIMARY KEY (fk\_branch\_br\_id, fk\_transaction\_transc\_id),

10 CONSTRAINT fk\_branch\_par FOREIGN KEY (fk\_branch\_br\_id) REFERENCES branch(br\_id) ON DELETE CASCADE,

11 CONSTRAINT fk\_transaction FOREIGN KEY (fk\_transaction\_transc\_id) REFERENCES transaction(transc\_id) ON DELETE CASCADE

12 );

Table OPERATION created.

SQL>

SQL> CREATE TABLE employee

2 ( emp\_id NUMBER(10) NOT NULL,

3 f\_name VARCHAR2(50) NOT NULL,

4 l\_name VARCHAR2(50) NOT NULL,

5 email VARCHAR2(50) NOT NULL,

6 position VARCHAR2(50) NOT NULL,

7 hired\_date DATE NOT NULL,

8 fk\_branch\_br\_id NUMBER(10),

9 CONSTRAINT employee\_pk PRIMARY KEY (emp\_id),

10 CONSTRAINT fk\_branch FOREIGN KEY (fk\_branch\_br\_id) REFERENCES branch(br\_id)

11 ON DELETE CASCADE

12 );

Table EMPLOYEE created.

SQL>

SQL>

SQL> --------------------------------------------------------------------------------

SQL> ----- 3. Create indexes for possible natural key, foreign key columns and others

SQL>

SQL> -- Create unique index on natural key columns.

SQL> CREATE UNIQUE INDEX idx\_emp\_fname\_lname ON employee(f\_name, l\_name);

INDEX IDX\_EMP\_FNAME\_LNAME created.

SQL> CREATE UNIQUE INDEX idx\_cust\_fname\_lname ON customer(f\_name, l\_name);

INDEX IDX\_CUST\_FNAME\_LNAME created.

SQL> -- Create index on foreign key columns.

SQL> CREATE INDEX idx\_acct\_cust\_id ON account(fk\_customer\_cust\_id);

Index IDX\_ACCT\_CUST\_ID created.

SQL> CREATE INDEX idx\_transc\_acct\_id ON transaction(fk\_account\_account\_id);

Index IDX\_TRANSC\_ACCT\_ID created.

SQL> CREATE INDEX idx\_op\_transc\_id ON operation(fk\_transaction\_transc\_id);

Index IDX\_OP\_TRANSC\_ID created.

SQL> CREATE INDEX idx\_op\_br\_id ON operation(fk\_branch\_br\_id);

Index IDX\_OP\_BR\_ID created.

SQL> CREATE INDEX idx\_emp\_br\_id ON employee(fk\_branch\_br\_id);

Index IDX\_EMP\_BR\_ID created.

SQL> -- Create index on other columns that will be frequently used as query filters

SQL> CREATE INDEX idx\_balance ON account(balance);

Index IDX\_BALANCE created.

SQL> CREATE INDEX idx\_transc\_type ON transaction(transc\_type);

Index IDX\_TRANSC\_TYPE created.

SQL> CREATE INDEX idx\_op\_created\_by ON operation(created\_by);

Index IDX\_OP\_CREATED\_BY created.

SQL>

SQL>

SQL> ---------------------------------------------------------------------------------

SQL> ----- 4. Create views that are appropriate to support business requirements -----

SQL> /\*

SQL>Business purpose: This view simplifies access to all accounts that are currently active. It is useful for operational dashboards and reporting

SQL>\*/

SQL> -- View showing all accounts with 'Active' status

SQL> CREATE OR REPLACE VIEW view\_active\_acct AS

2 SELECT account\_id, fk\_customer\_cust\_id, acct\_type, balance, creat\_date

3 FROM account

4 Where status = 'Active';

View VIEW\_ACTIVE\_ACCT created.

SQL>

SQL> /\*

SQL>Business purpose: This view provides a consolidated report of the customer details and the sum of all their account balances.

SQL> It is useful for customer and financial analysis

SQL>\*/

SQL> -- This view combines customer details with their account balances

SQL> CREATE OR REPLACE VIEW view\_cust\_acct\_summary AS

2 SELECT c.cust\_id, c.f\_name, c.l\_name, c.email, c.ph\_number, NVL(SUM(a.balance), 0) AS total\_balance

3 FROM customer c LEFT JOIN account a ON c.cust\_id = a.fk\_customer\_cust\_id

4 GROUP BY c.cust\_id, c.f\_name, c.l\_name, c.email, c.ph\_number;

View VIEW\_CUST\_ACCT\_SUMMARY created.

SQL>

SQL> /\*

SQL> Business purpose: This view provides crucial information about each branch. It will help allocate resources proportional

SQL> Also, it is useful to further understand the causes of frequentation to make better decision.

SQL> \*/

SQL> -- This view consist to know the frequentation of each branch by customers

SQL> CREATE OR REPLACE VIEW view\_br\_performance AS

2 SELECT t.location, t.br\_name, t.channel, COUNT(t.location) AS number\_of\_frequention

3 FROM (

4 SELECT b.location, b.br\_name, o.channel

5 FROM branch b INNER JOIN operation o ON b.br\_id = o.fk\_branch\_br\_id

6 WHERE o.channel IN ('Teller', 'ATM')) t

7 GROUP BY t.location, t.br\_name, t.channel;

View VIEW\_BR\_PERFORMANCE created.

SQL>

SQL>

SQL> ------------------------------------------------------------------------------------

SQL> ----- 5. Create sequences for the primary keys created in each entities ------------

SQL>

SQL> -- Generate values for cust\_id (primary key of customer entity) through sequence

SQL> CREATE SEQUENCE seq\_cust\_id

2 MINVALUE 1

3 MAXVALUE 100

4 START WITH 1

5 INCREMENT BY 1

6 CACHE 20;

Sequence SEQ\_CUST\_ID created.

SQL>

SQL> -- Generate values for br\_id (primary key of branch entity) through sequence

SQL> CREATE SEQUENCE seq\_br\_id

2 MINVALUE 101

3 MAXVALUE 200

4 START WITH 101

5 INCREMENT BY 1

6 CACHE 20;

Sequence SEQ\_BR\_ID created.

SQL>

SQL> -- Generate values for emp\_id (primary key of employee entity) through sequence

SQL> CREATE SEQUENCE seq\_emp\_id

2 MINVALUE 201

3 MAXVALUE 400

4 START WITH 201

5 INCREMENT BY 1

6 CACHE 20;

Sequence SEQ\_EMP\_ID created.

SQL>

SQL> -- Generate values for account\_id (primary key of account entity) through sequence

SQL> CREATE SEQUENCE seq\_acct\_id

2 MINVALUE 401

3 MAXVALUE 600

4 START WITH 401

5 INCREMENT BY 1

6 CACHE 20;

Sequence SEQ\_ACCT\_ID created.

SQL>

SQL> -- Generate values for transc\_id (primary key transaction entity) through sequence

SQL> CREATE SEQUENCE seq\_transc\_id

2 MINVALUE 601

3 MAXVALUE 800

4 START WITH 601

5 INCREMENT BY 1

6 CACHE 20;

Sequence SEQ\_TRANSC\_ID created.

SQL>

SQL>

SQL> ------------------------------------------------------------------------------------

SQL> ----- 6. Create triggers ----------------------------------------------------------

SQL> CREATE OR REPLACE TRIGGER trigger\_cust\_id

2 BEFORE INSERT ON customer

3 FOR EACH ROW

4 BEGIN

5 IF :NEW.cust\_id IS NULL THEN

6 :NEW.cust\_id := seq\_cust\_id.NEXTVAL;

7 END IF;

8 END;

9 /

Trigger TRIGGER\_CUST\_ID compiled

SQL>

SQL> CREATE OR REPLACE TRIGGER trigger\_acct\_id

2 BEFORE INSERT ON account

3 FOR EACH ROW

4 BEGIN

5 IF :NEW.account\_id IS NULL THEN

6 :NEW.account\_id := seq\_acct\_id.NEXTVAL;

7 END IF;

8 END;

9 /

Trigger TRIGGER\_ACCT\_ID compiled

SQL>

SQL> CREATE OR REPLACE TRIGGER trigger\_emp\_id

2 BEFORE INSERT ON employee

3 FOR EACH ROW

4 BEGIN

5 IF :NEW.emp\_id IS NULL THEN

6 :NEW.emp\_id := seq\_emp\_id.NEXTVAL;

7 END IF;

8 END;

9 /

Trigger TRIGGER\_EMP\_ID compiled

SQL>

SQL> CREATE OR REPLACE TRIGGER trigger\_br\_id

2 BEFORE INSERT ON branch

3 FOR EACH ROW

4 BEGIN

5 IF :NEW.br\_id IS NULL THEN

6 :NEW.br\_id := seq\_br\_id.NEXTVAL;

7 END IF;

8 END;

9 /

Trigger TRIGGER\_BR\_ID compiled

SQL>

SQL> CREATE OR REPLACE TRIGGER trigger\_transc\_id

2 BEFORE INSERT ON transaction

3 FOR EACH ROW

4 BEGIN

5 IF :NEW.transc\_id IS NULL THEN

6 :NEW.transc\_id := seq\_transc\_id.NEXTVAL;

7 END IF;

8 END;

9 /

Trigger TRIGGER\_TRANSC\_ID compiled

SQL>

SQL> /\* The business value of this trigger is to ensure auditability for transaction records.

SQL>-- It automatically logs the staff who performed the transaction and when, helping meet compliance and tracking requirements.

SQL>\*/

SQL> CREATE OR REPLACE TRIGGER trg\_operation\_audit

2 BEFORE INSERT ON operation

3 FOR EACH ROW

4 BEGIN

5 :NEW.date\_created := CAST(SYSDATE AS TIMESTAMP);

6 END;

7 /

Trigger TRG\_OPERATION\_AUDIT compiled

SQL>

SQL> /\* The business value of this trigger is to ensure auditability of when the customer initiates the transaction.

SQL> It also ensures that the bank meet compliance and tracking requirements.

SQL>\*/

SQL> CREATE OR REPLACE TRIGGER trg\_transaction\_audit

2 BEFORE INSERT ON Transaction

3 FOR EACH ROW

4 BEGIN

5 :NEW.timestamp := CAST(SYSDATE AS TIMESTAMP);

6 END;

7 /

Trigger TRG\_TRANSACTION\_AUDIT compiled

SQL>

SQL>

SQL> -- To confirm all your tables are present

SQL> SELECT TABLE\_NAME FROM USER\_TABLES;

TABLE\_NAME

--------------------------------------------------------------------------------------------------------------------------------

COURSE

ENROLLMENT

GRADE

GRADE\_CONVERSION

GRADE\_TYPE

GRADE\_TYPE\_WEIGHT

INSTRUCTOR

SECTION

STUDENT

ZIPCODE

OPERATION

TABLE\_NAME

--------------------------------------------------------------------------------------------------------------------------------

EMPLOYEE

CUSTOMER

ACCOUNT

TRANSACTION

BRANCH

16 rows selected.

SQL>

SQL> -- To check the status and types of all your created objects (valid/invalid)

SQL> SELECT OBJECT\_NAME, STATUS, CREATED, LAST\_DDL\_TIME FROM USER\_OBJECTS;

OBJECT\_NAME STATUS CREATED LAST\_DDL\_

-------------------------------------------------------------------------------------------------------------------------------- ------- --------- ---------

INSTRUCTOR\_ID\_SEQ VALID 16-FEB-22 16-FEB-22

SECTION\_ID\_SEQ VALID 16-FEB-22 16-FEB-22

STUDENT\_ID\_SEQ VALID 16-FEB-22 16-FEB-22

COURSE\_NO\_SEQ VALID 16-FEB-22 16-FEB-22

COURSE VALID 16-FEB-22 16-FEB-22

CRSE\_CRSE\_FK\_I VALID 16-FEB-22 16-FEB-22

CRSE\_PK VALID 16-FEB-22 16-FEB-22

ENROLLMENT VALID 16-FEB-22 16-FEB-22

ENR\_SECT\_FK\_I VALID 16-FEB-22 16-FEB-22

ENR\_PK VALID 16-FEB-22 16-FEB-22

GRADE VALID 16-FEB-22 16-FEB-22

OBJECT\_NAME STATUS CREATED LAST\_DDL\_

-------------------------------------------------------------------------------------------------------------------------------- ------- --------- ---------

GR\_GRTW\_FK\_I VALID 16-FEB-22 16-FEB-22

GR\_PK VALID 16-FEB-22 16-FEB-22

GRADE\_CONVERSION VALID 16-FEB-22 16-FEB-22

GRCON\_PK VALID 16-FEB-22 16-FEB-22

GRADE\_TYPE VALID 16-FEB-22 16-FEB-22

GRTYP\_PK VALID 16-FEB-22 16-FEB-22

GRADE\_TYPE\_WEIGHT VALID 16-FEB-22 16-FEB-22

GRTW\_GRTYP\_FK\_I VALID 16-FEB-22 16-FEB-22

GRTW\_PK VALID 16-FEB-22 16-FEB-22

INSTRUCTOR VALID 16-FEB-22 16-FEB-22

INST\_ZIP\_FK\_I VALID 16-FEB-22 16-FEB-22

OBJECT\_NAME STATUS CREATED LAST\_DDL\_

-------------------------------------------------------------------------------------------------------------------------------- ------- --------- ---------

INST\_PK VALID 16-FEB-22 16-FEB-22

SECTION VALID 16-FEB-22 16-FEB-22

SECT\_CRSE\_FK\_I VALID 16-FEB-22 16-FEB-22

SECT\_INST\_FK\_I VALID 16-FEB-22 16-FEB-22

SECT\_PK VALID 16-FEB-22 16-FEB-22

SECT\_SECT2\_UK VALID 16-FEB-22 16-FEB-22

STUDENT VALID 16-FEB-22 16-FEB-22

STU\_ZIP\_FK\_I VALID 16-FEB-22 16-FEB-22

STU\_PK VALID 16-FEB-22 16-FEB-22

ZIPCODE VALID 16-FEB-22 16-FEB-22

ZIP\_PK VALID 16-FEB-22 16-FEB-22

OBJECT\_NAME STATUS CREATED LAST\_DDL\_

-------------------------------------------------------------------------------------------------------------------------------- ------- --------- ---------

VIEW\_ACTIVE\_ACCT VALID 14-JUL-25 05-AUG-25

OPERATION\_PK VALID 05-AUG-25 05-AUG-25

OPERATION VALID 05-AUG-25 05-AUG-25

EMPLOYEE VALID 05-AUG-25 05-AUG-25

EMPLOYEE\_PK VALID 05-AUG-25 05-AUG-25

IDX\_EMP\_FNAME\_LNAME VALID 05-AUG-25 05-AUG-25

IDX\_CUST\_FNAME\_LNAME VALID 05-AUG-25 05-AUG-25

IDX\_ACCT\_CUST\_ID VALID 05-AUG-25 05-AUG-25

IDX\_TRANSC\_ACCT\_ID VALID 05-AUG-25 05-AUG-25

IDX\_OP\_TRANSC\_ID VALID 05-AUG-25 05-AUG-25

IDX\_OP\_BR\_ID VALID 05-AUG-25 05-AUG-25

OBJECT\_NAME STATUS CREATED LAST\_DDL\_

-------------------------------------------------------------------------------------------------------------------------------- ------- --------- ---------

IDX\_EMP\_BR\_ID VALID 05-AUG-25 05-AUG-25

IDX\_BALANCE VALID 05-AUG-25 05-AUG-25

IDX\_TRANSC\_TYPE VALID 05-AUG-25 05-AUG-25

IDX\_OP\_CREATED\_BY VALID 05-AUG-25 05-AUG-25

SEQ\_CUST\_ID VALID 05-AUG-25 05-AUG-25

SEQ\_BR\_ID VALID 05-AUG-25 05-AUG-25

SEQ\_EMP\_ID VALID 05-AUG-25 05-AUG-25

SEQ\_ACCT\_ID VALID 05-AUG-25 05-AUG-25

SEQ\_TRANSC\_ID VALID 05-AUG-25 05-AUG-25

VIEW\_CUST\_ACCT\_SUMMARY VALID 14-JUL-25 05-AUG-25

VIEW\_BR\_PERFORMANCE VALID 14-JUL-25 05-AUG-25

OBJECT\_NAME STATUS CREATED LAST\_DDL\_

-------------------------------------------------------------------------------------------------------------------------------- ------- --------- ---------

TRG\_OPERATION\_AUDIT VALID 05-AUG-25 05-AUG-25

TRIGGER\_EMP\_ID VALID 05-AUG-25 05-AUG-25

TRIGGER\_BR\_ID VALID 05-AUG-25 05-AUG-25

TRIGGER\_TRANSC\_ID VALID 05-AUG-25 05-AUG-25

TRG\_TRANSACTION\_AUDIT VALID 05-AUG-25 05-AUG-25

TRIGGER\_ACCT\_ID VALID 05-AUG-25 05-AUG-25

TRIGGER\_CUST\_ID VALID 05-AUG-25 05-AUG-25

CUSTOMER VALID 05-AUG-25 05-AUG-25

CUSTOMER\_PK VALID 05-AUG-25 05-AUG-25

ACCOUNT VALID 05-AUG-25 05-AUG-25

ACCOUNT\_PK VALID 05-AUG-25 05-AUG-25

OBJECT\_NAME STATUS CREATED LAST\_DDL\_

-------------------------------------------------------------------------------------------------------------------------------- ------- --------- ---------

TRANSACTION VALID 05-AUG-25 05-AUG-25

TRANSACTION\_PK VALID 05-AUG-25 05-AUG-25

BRANCH VALID 05-AUG-25 05-AUG-25

BRANCH\_PK VALID 05-AUG-25 05-AUG-25

70 rows selected.

-- DML Output --

SQL> SQL> set echo on; SQL> SQL> -- DML- SQL INSERT statement that populate CUSTOMER table withn sample data. CUSTOMER data consists of 10 rows. SQL> INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Alicia', 'Mills', TO\_DATE('1991-05-24', 'YYYY-MM-DD'), '19 Garden Ave, Austin', '5125551001', '[alicia.mills@example.com](mailto:alicia.mills@example.com)');

1 row inserted.

SQL> INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Brian', 'Tucker', TO\_DATE('1988-11-07', 'YYYY-MM-DD'), '82 Ridge Rd, Denver', '3035554521', '[brian.tucker@example.com](mailto:brian.tucker@example.com)');

1 row inserted.

SQL> INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Clara', 'Nguyen', TO\_DATE('1995-02-16', 'YYYY-MM-DD'), '34 Willow St, Seattle', '2065553122', '[clara.nguyen@example.com](mailto:clara.nguyen@example.com)');

1 row inserted.

SQL> INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Derek', 'Ford', TO\_DATE('1990-09-12', 'YYYY-MM-DD'), '106 Main Blvd, Miami', '7865559982', '[derek.ford@example.com](mailto:derek.ford@example.com)');

1 row inserted.

SQL> INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Ella', 'Brown', TO\_DATE('1986-03-04', 'YYYY-MM-DD'), '501 Pacific St, Portland', '5035557411', '[ella.brown@example.com](mailto:ella.brown@example.com)');

1 row inserted.

SQL> INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Frank', 'Lopez', TO\_DATE('1993-07-19', 'YYYY-MM-DD'), '212 Walnut Ln, Chicago', '7735552188', '[frank.lopez@example.com](mailto:frank.lopez@example.com)');

1 row inserted.

SQL> INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Grace', 'Kim', TO\_DATE('1989-12-02', 'YYYY-MM-DD'), '45 Sunset Ave, LA', '2135553011', '[grace.kim@example.com](mailto:grace.kim@example.com)');

1 row inserted.

SQL> INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Henry', 'Singh', TO\_DATE('1992-06-28', 'YYYY-MM-DD'), '87 Liberty St, Boston', '6175559003', '[henry.singh@example.com](mailto:henry.singh@example.com)');

1 row inserted.

SQL> INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Isla', 'Patel', TO\_DATE('1994-08-10', 'YYYY-MM-DD'), '15 Beacon Way, New York', '7185556020', '[isla.patel@example.com](mailto:isla.patel@example.com)');

1 row inserted.

SQL> INSERT INTO CUSTOMER (f\_name, l\_name, dob, address, ph\_number, email) VALUES ('Jack', 'Davis', TO\_DATE('1996-10-05', 'YYYY-MM-DD'), '68 Pine Circle, Dallas', '2145557777', '[jack.davis@example.com](mailto:jack.davis@example.com)');

1 row inserted.

SQL> SQL> -- DML- SQL INSERT statement that populate ACCOUNT table withn sample data. ACCOUNT data consists of 25 rows. SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (2300.50, 'Savings', TO\_DATE('2022-01-10', 'YYYY-MM-DD'), 'Active', 1);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (150.75, 'Checking', TO\_DATE('2022-03-22', 'YYYY-MM-DD'), 'Active', 2);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (6700.00, 'Loan', TO\_DATE('2021-09-05', 'YYYY-MM-DD'), 'Closed', 3);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (980.20, 'Savings', TO\_DATE('2022-07-13', 'YYYY-MM-DD'), 'Active', 4);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (3120.00, 'Checking', TO\_DATE('2023-01-01', 'YYYY-MM-DD'), 'Active', 5);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (4900.00, 'Loan', TO\_DATE('2021-12-15', 'YYYY-MM-DD'), 'Delinquent', 6);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (1125.35, 'Savings', TO\_DATE('2022-04-22', 'YYYY-MM-DD'), 'Active', 7);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (305.60, 'Checking', TO\_DATE('2022-08-09', 'YYYY-MM-DD'), 'Frozen', 8);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (15000.00, 'Loan', TO\_DATE('2021-06-21', 'YYYY-MM-DD'), 'Active', 9);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (220.10, 'Savings', TO\_DATE('2023-03-11', 'YYYY-MM-DD'), 'Active', 10);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (705.90, 'Checking', TO\_DATE('2022-02-28', 'YYYY-MM-DD'), 'Active', 1);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (9900.00, 'Loan', TO\_DATE('2022-05-15', 'YYYY-MM-DD'), 'Active', 2);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (650.00, 'Savings', TO\_DATE('2023-06-20', 'YYYY-MM-DD'), 'Active', 3);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (120.00, 'Checking', TO\_DATE('2022-09-01', 'YYYY-MM-DD'), 'Frozen', 4);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (4200.00, 'Loan', TO\_DATE('2021-03-03', 'YYYY-MM-DD'), 'Closed', 5);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (1580.30, 'Savings', TO\_DATE('2023-04-12', 'YYYY-MM-DD'), 'Active', 2);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (875.40, 'Checking', TO\_DATE('2022-10-05', 'YYYY-MM-DD'), 'Active', 4);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (2400.00, 'Loan', TO\_DATE('2020-12-01', 'YYYY-MM-DD'), 'Active', 6);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (130.20, 'Savings', TO\_DATE('2023-07-01', 'YYYY-MM-DD'), 'Active', 8);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (395.90, 'Checking', TO\_DATE('2021-08-30', 'YYYY-MM-DD'), 'Closed', 9);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (7600.00, 'Loan', TO\_DATE('2023-05-23', 'YYYY-MM-DD'), 'Active', 10);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (999.99, 'Savings', TO\_DATE('2022-11-18', 'YYYY-MM-DD'), 'Active', 2);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (543.21, 'Checking', TO\_DATE('2023-02-14', 'YYYY-MM-DD'), 'Frozen', 3);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (8200.00, 'Loan', TO\_DATE('2021-01-01', 'YYYY-MM-DD'), 'Delinquent', 5);

1 row inserted.

SQL> INSERT INTO ACCOUNT (balance, acct\_type, creat\_date, status, fk\_customer\_cust\_id) VALUES (510.00, 'Savings', TO\_DATE('2023-08-08', 'YYYY-MM-DD'), 'Active', 1);

1 row inserted.

SQL> SQL> -- DML- SQL INSERT statement that populate TRANSACTION table withn sample data. TRANSACTION data consists of 15 rows. SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (200.00, 'Deposit', 'Initial deposit', 401);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES(100.00, 'Withdrawal', 'ATM withdrawal', 402);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (300.00, 'Transfer', 'Transfer to savings', 403);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (50.00, 'Withdrawal', 'POS Purchase', 404);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (1000.00, 'Deposit', 'Salary', 405);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (25.00, 'Withdrawal', 'ATM fee', 403);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (500.00, 'Deposit', 'Freelance payment', 407);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (75.00, 'Transfer', 'Transfer to friend', 408);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (120.00, 'Deposit', 'Cash deposit', 403);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (200.00, 'Withdrawal', 'Bill payment', 410);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (500.00, 'Deposit', 'Initial deposit', 415);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (150.00, 'Withdrawal', 'ATM withdrawal', 423);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (300.00, 'Transfer', 'Transfer to savings', 420);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (75.00, 'Withdrawal', 'POS purchase', 423);

1 row inserted.

SQL> INSERT INTO TRANSACTION (amount, transc\_type, description, fk\_account\_account\_id) VALUES (600.00, 'Transfer', 'Loan repayment', 425);

1 row inserted.

SQL> SQL> -- DML- SQL INSERT statement that populate BRANCH table withn sample data. BRANCH data consists of 10 rows. SQL> INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Downtown', 'New York', 'Alice Miles', '2125551010', TO\_DATE('2010-01-01','YYYY-MM-DD'));

1 row inserted.

SQL> INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Midtown', 'Chicago', 'Brian Lee', '3125551020', TO\_DATE('2012-05-15','YYYY-MM-DD'));

1 row inserted.

SQL> INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Uptown', 'Houston', 'Carla Yu', '7135551030', TO\_DATE('2014-03-20','YYYY-MM-DD'));

1 row inserted.

SQL> INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Central', 'Los Angeles', 'David Singh', '2135551040', TO\_DATE('2011-08-10','YYYY-MM-DD'));

1 row inserted.

SQL> INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Westside', 'San Diego', 'Eva Knight', '6195551050', TO\_DATE('2013-09-01','YYYY-MM-DD'));

1 row inserted.

SQL> INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Southpoint', 'Miami', 'Fred Owen', '3055551060', TO\_DATE('2015-10-12','YYYY-MM-DD'));

1 row inserted.

SQL> INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Riverside', 'Denver', 'Grace Patel', '3035551070', TO\_DATE('2016-07-21','YYYY-MM-DD'));

1 row inserted.

SQL> INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Northend', 'Seattle', 'Henry Kim', '2065551080', TO\_DATE('2018-02-14','YYYY-MM-DD'));

1 row inserted.

SQL> INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Eastside', 'Atlanta', 'Isla Brown', '4045551090', TO\_DATE('2017-04-25','YYYY-MM-DD'));

1 row inserted.

SQL> INSERT INTO BRANCH (br\_name, location, mgr\_name, ph\_number, est\_date) VALUES ('Metro', 'Boston', 'Jack Chen', '6175551100', TO\_DATE('2019-12-31','YYYY-MM-DD'));

1 row inserted.

SQL> SQL> -- DML- SQL INSERT statement that populate EMPLOYEE table withn sample data. EMPLOYEE data consists of 10 rows. SQL> INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Angela', 'White', '[awhite@example.com](mailto:awhite@example.com)', 'Teller', TO\_DATE('2018-04-01','YYYY-MM-DD'), 101);

1 row inserted.

SQL> INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES('Ben', 'King', '[bking@example.com](mailto:bking@example.com)', 'Manager', TO\_DATE('2017-03-15','YYYY-MM-DD'), 102);

1 row inserted.

SQL> INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Chris', 'Davis', '[cdavis@example.com](mailto:cdavis@example.com)', 'Clerk', TO\_DATE('2019-06-22','YYYY-MM-DD'), 103);

1 row inserted.

SQL> INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Diana', 'Morris', '[dmorris@example.com](mailto:dmorris@example.com)', 'Security', TO\_DATE('2020-01-10','YYYY-MM-DD'), 104);

1 row inserted.

SQL> INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Ethan', 'Scott', '[escott@example.com](mailto:escott@example.com)', 'Teller', TO\_DATE('2016-08-12','YYYY-MM-DD'), 105);

1 row inserted.

SQL> INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Fiona', 'Hill', '[fhill@example.com](mailto:fhill@example.com)', 'Clerk', TO\_DATE('2015-02-27','YYYY-MM-DD'), 106);

1 row inserted.

SQL> INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('George', 'Young', '[gyoung@example.com](mailto:gyoung@example.com)', 'Manager', TO\_DATE('2014-11-03','YYYY-MM-DD'), 107);

1 row inserted.

SQL> INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Hannah', 'Green', '[hgreen@example.com](mailto:hgreen@example.com)', 'Clerk', TO\_DATE('2021-05-18','YYYY-MM-DD'), 108);

1 row inserted.

SQL> INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Ian', 'Lopez', '[ilopez@example.com](mailto:ilopez@example.com)', 'Teller', TO\_DATE('2013-09-09','YYYY-MM-DD'), 109);

1 row inserted.

SQL> INSERT INTO EMPLOYEE (f\_name, l\_name, email, position, hired\_date, fk\_branch\_br\_id) VALUES ('Judy', 'Turner', '[jturner@example.com](mailto:jturner@example.com)', 'Manager', TO\_DATE('2012-12-12','YYYY-MM-DD'), 110);

1 row inserted.

SQL> SQL> -- DML- SQL INSERT statement that populate OPERATION table withn sample data. OPERATION data consists of 20 rows. SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Angela White', 'Teller', 'AP123', 'Approved', 101, 601);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Ben King', 'ATM', 'AP124', 'Completed', 102, 602);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Chris Davis', 'Online', 'AP125', 'Approved', 103, 603);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Diana Morris', 'Teller', 'AP126', 'Rejected', 104, 604);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Ethan Scott', 'ATM', 'AP127', 'Approved', 105, 605);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Fiona Hill', 'Online', 'AP128', 'Approved', 106, 606);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('George Young', 'Teller', 'AP129', 'Completed', 107, 607);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Hannah Green', 'ATM', 'AP130', 'Approved', 108, 608);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Ian Lopez', 'Online', 'AP131', 'Approved', 109, 609);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Judy Turner', 'Teller', 'AP132', 'Pending', 110, 610);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Angela White', 'ATM', 'AP133', 'Approved', 101, 611);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Ben King', 'Online', 'AP134', 'Completed', 102, 612);

1 row inserted.

SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Chris Davis', 'Teller', 'AP135', 'Approved', 103, 613);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Diana Morris', 'ATM', 'AP136', 'Completed', 104, 614);

1 row inserted.

SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Ethan Scott', 'Online', 'AP137', 'Approved', 105, 615);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Fiona Hill', 'Teller', 'AP138', 'Rejected', 106, 601);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('George Young', 'ATM', 'AP139', 'Pending', 107, 602);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Hannah Green', 'Online', 'AP140', 'Completed', 108, 603);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Ian Lopez', 'Teller', 'AP141', 'Approved', 109, 604);

1 row inserted.

SQL> -- SQL> INSERT INTO OPERATION (created\_by, channel, appr\_code, op\_status, fk\_branch\_br\_id, fk\_transaction\_transc\_id) VALUES ('Judy Turner', 'ATM', 'AP142', 'Approved', 110, 605);

1 row inserted.

SQL> -- SQL> SQL> commit;

Commit complete.

SQL> SQL> /\* SQL>20 SQL Queries : 12 Basic, 8 Advanced SQL>\*/ SQL> -- Query 1: select all columns and all rows from one table SQL> -- Business purpose: this query selects all information about all customers SQL> select \* from customer;

CUST\_ID F\_NAME L\_NAME DOB ADDRESS PH\_NUMBER EMAIL

1 Alicia Mills 24-MAY-91 19 Garden Ave, Austin 5125551001 [alicia.mills@example.com](mailto:alicia.mills@example.com)   
 2 Brian Tucker 07-NOV-88 82 Ridge Rd, Denver 3035554521 [brian.tucker@example.com](mailto:brian.tucker@example.com)   
 3 Clara Nguyen 16-FEB-95 34 Willow St, Seattle 2065553122 [clara.nguyen@example.com](mailto:clara.nguyen@example.com)   
 4 Derek Ford 12-SEP-90 106 Main Blvd, Miami 7865559982 [derek.ford@example.com](mailto:derek.ford@example.com)   
 5 Ella Brown 04-MAR-86 501 Pacific St, Portland 5035557411 [ella.brown@example.com](mailto:ella.brown@example.com)   
 6 Frank Lopez 19-JUL-93 212 Walnut Ln, Chicago 7735552188 [frank.lopez@example.com](mailto:frank.lopez@example.com)   
 7 Grace Kim 02-DEC-89 45 Sunset Ave, LA 2135553011 [grace.kim@example.com](mailto:grace.kim@example.com)   
 8 Henry Singh 28-JUN-92 87 Liberty St, Boston 6175559003 [henry.singh@example.com](mailto:henry.singh@example.com)   
 9 Isla Patel 10-AUG-94 15 Beacon Way, New York 7185556020 [isla.patel@example.com](mailto:isla.patel@example.com)   
 10 Jack Davis 05-OCT-96 68 Pine Circle, Dallas 2145557777 [jack.davis@example.com](mailto:jack.davis@example.com)

10 rows selected.

SQL> SQL> -- Query 2: Select five columns and all rows from one table SQL> -- Business purpose: this query selects all information from five attributes about all branch SQL> select br\_name as "Branch Name", location as "Location", mgr\_name as "Manager Name", ph\_number as "Phone Number", est\_date as "Establishment Date" from branch;

Branch Name Location Manager Name Phone Numb Establish

Downtown New York Alice Miles 2125551010 01-JAN-10 Midtown Chicago Brian Lee 3125551020 15-MAY-12 Uptown Houston Carla Yu 7135551030 20-MAR-14 Central Los Angeles David Singh 2135551040 10-AUG-11 Westside San Diego Eva Knight 6195551050 01-SEP-13 Southpoint Miami Fred Owen 3055551060 12-OCT-15 Riverside Denver Grace Patel 3035551070 21-JUL-16 Northend Seattle Henry Kim 2065551080 14-FEB-18 Eastside Atlanta Isla Brown 4045551090 25-APR-17 Metro Boston Jack Chen 6175551100 31-DEC-19

10 rows selected.

SQL> SQL> --Query 3: Select all columns from all rows from one view SQL> -- Business purpose: this selects all information about a created view called view\_active\_acct SQL> select \* from view\_active\_acct;

ACCOUNT\_ID FK\_CUSTOMER\_CUST\_ID ACCT\_TYPE BALANCE CREAT\_DAT

401 1 Savings 2300.5 10-JAN-22  
 402 2 Checking 150.75 22-MAR-22  
 404 4 Savings 980.2 13-JUL-22  
 405 5 Checking 3120 01-JAN-23  
 407 7 Savings 1125.35 22-APR-22  
 409 9 Loan 15000 21-JUN-21  
 410 10 Savings 220.1 11-MAR-23  
 411 1 Checking 705.9 28-FEB-22  
 412 2 Loan 9900 15-MAY-22  
 413 3 Savings 650 20-JUN-23  
 416 2 Savings 1580.3 12-APR-23

ACCOUNT\_ID FK\_CUSTOMER\_CUST\_ID ACCT\_TYPE BALANCE CREAT\_DAT

417 4 Checking 875.4 05-OCT-22  
 418 6 Loan 2400 01-DEC-20  
 419 8 Savings 130.2 01-JUL-23  
 421 10 Loan 7600 23-MAY-23  
 422 2 Savings 999.99 18-NOV-22  
 425 1 Savings 510 08-AUG-23

17 rows selected.

SQL> SQL> --Query 4: Using a join on 2 tables, select all columns and all rows from the tables without the use of a Cartesian product SQL> -- Business purpose: this query joins all information from two table where there is a matching value on the customer id SQL> select \* from customer c inner join account a on c.cust\_id = a.fk\_customer\_cust\_id;

CUST\_ID F\_NAME L\_NAME DOB ADDRESS PH\_NUMBER EMAIL ACCOUNT\_ID BALANCE ACCT\_TYPE CREAT\_DAT STATUS FK\_CUSTOMER\_CUST\_ID

1 Alicia Mills 24-MAY-91 19 Garden Ave, Austin 5125551001 [alicia.mills@example.com](mailto:alicia.mills@example.com) 401 2300.5 Savings 10-JAN-22 Active 1  
 1 Alicia Mills 24-MAY-91 19 Garden Ave, Austin 5125551001 [alicia.mills@example.com](mailto:alicia.mills@example.com) 411 705.9 Checking 28-FEB-22 Active 1  
 1 Alicia Mills 24-MAY-91 19 Garden Ave, Austin 5125551001 [alicia.mills@example.com](mailto:alicia.mills@example.com) 425 510 Savings 08-AUG-23 Active 1  
 2 Brian Tucker 07-NOV-88 82 Ridge Rd, Denver 3035554521 [brian.tucker@example.com](mailto:brian.tucker@example.com) 402 150.75 Checking 22-MAR-22 Active 2  
 2 Brian Tucker 07-NOV-88 82 Ridge Rd, Denver 3035554521 [brian.tucker@example.com](mailto:brian.tucker@example.com) 412 9900 Loan 15-MAY-22 Active 2  
 2 Brian Tucker 07-NOV-88 82 Ridge Rd, Denver 3035554521 [brian.tucker@example.com](mailto:brian.tucker@example.com) 416 1580.3 Savings 12-APR-23 Active 2  
 2 Brian Tucker 07-NOV-88 82 Ridge Rd, Denver 3035554521 [brian.tucker@example.com](mailto:brian.tucker@example.com) 422 999.99 Savings 18-NOV-22 Active 2  
 3 Clara Nguyen 16-FEB-95 34 Willow St, Seattle 2065553122 [clara.nguyen@example.com](mailto:clara.nguyen@example.com) 403 6700 Loan 05-SEP-21 Closed 3  
 3 Clara Nguyen 16-FEB-95 34 Willow St, Seattle 2065553122 [clara.nguyen@example.com](mailto:clara.nguyen@example.com) 413 650 Savings 20-JUN-23 Active 3  
 3 Clara Nguyen 16-FEB-95 34 Willow St, Seattle 2065553122 [clara.nguyen@example.com](mailto:clara.nguyen@example.com) 423 543.21 Checking 14-FEB-23 Frozen 3  
 4 Derek Ford 12-SEP-90 106 Main Blvd, Miami 7865559982 [derek.ford@example.com](mailto:derek.ford@example.com) 404 980.2 Savings 13-JUL-22 Active 4

CUST\_ID F\_NAME L\_NAME DOB ADDRESS PH\_NUMBER EMAIL ACCOUNT\_ID BALANCE ACCT\_TYPE CREAT\_DAT STATUS FK\_CUSTOMER\_CUST\_ID

4 Derek Ford 12-SEP-90 106 Main Blvd, Miami 7865559982 [derek.ford@example.com](mailto:derek.ford@example.com) 414 120 Checking 01-SEP-22 Frozen 4  
 4 Derek Ford 12-SEP-90 106 Main Blvd, Miami 7865559982 [derek.ford@example.com](mailto:derek.ford@example.com) 417 875.4 Checking 05-OCT-22 Active 4  
 5 Ella Brown 04-MAR-86 501 Pacific St, Portland 5035557411 [ella.brown@example.com](mailto:ella.brown@example.com) 405 3120 Checking 01-JAN-23 Active 5  
 5 Ella Brown 04-MAR-86 501 Pacific St, Portland 5035557411 [ella.brown@example.com](mailto:ella.brown@example.com) 415 4200 Loan 03-MAR-21 Closed 5  
 5 Ella Brown 04-MAR-86 501 Pacific St, Portland 5035557411 [ella.brown@example.com](mailto:ella.brown@example.com) 424 8200 Loan 01-JAN-21 Delinquent 5  
 6 Frank Lopez 19-JUL-93 212 Walnut Ln, Chicago 7735552188 [frank.lopez@example.com](mailto:frank.lopez@example.com) 406 4900 Loan 15-DEC-21 Delinquent 6  
 6 Frank Lopez 19-JUL-93 212 Walnut Ln, Chicago 7735552188 [frank.lopez@example.com](mailto:frank.lopez@example.com) 418 2400 Loan 01-DEC-20 Active 6  
 7 Grace Kim 02-DEC-89 45 Sunset Ave, LA 2135553011 [grace.kim@example.com](mailto:grace.kim@example.com) 407 1125.35 Savings 22-APR-22 Active 7  
 8 Henry Singh 28-JUN-92 87 Liberty St, Boston 6175559003 [henry.singh@example.com](mailto:henry.singh@example.com) 408 305.6 Checking 09-AUG-22 Frozen 8  
 8 Henry Singh 28-JUN-92 87 Liberty St, Boston 6175559003 [henry.singh@example.com](mailto:henry.singh@example.com) 419 130.2 Savings 01-JUL-23 Active 8  
 9 Isla Patel 10-AUG-94 15 Beacon Way, New York 7185556020 [isla.patel@example.com](mailto:isla.patel@example.com) 409 15000 Loan 21-JUN-21 Active 9

CUST\_ID F\_NAME L\_NAME DOB ADDRESS PH\_NUMBER EMAIL ACCOUNT\_ID BALANCE ACCT\_TYPE CREAT\_DAT STATUS FK\_CUSTOMER\_CUST\_ID

9 Isla Patel 10-AUG-94 15 Beacon Way, New York 7185556020 [isla.patel@example.com](mailto:isla.patel@example.com) 420 395.9 Checking 30-AUG-21 Closed 9  
 10 Jack Davis 05-OCT-96 68 Pine Circle, Dallas 2145557777 [jack.davis@example.com](mailto:jack.davis@example.com) 410 220.1 Savings 11-MAR-23 Active 10  
 10 Jack Davis 05-OCT-96 68 Pine Circle, Dallas 2145557777 [jack.davis@example.com](mailto:jack.davis@example.com) 421 7600 Loan 23-MAY-23 Active 10

25 rows selected.

SQL> SQL> --Query 5: Select and order data retrieved from one table SQL> -- Business purpose: this query select all information of all employee in ascending order based on employee name SQL> select \* from employee order by f\_name, l\_name asc;

EMP\_ID F\_NAME L\_NAME EMAIL POSITION HIRED\_DAT FK\_BRANCH\_BR\_ID

201 Angela White [awhite@example.com](mailto:awhite@example.com) Teller 01-APR-18 101  
 202 Ben King [bking@example.com](mailto:bking@example.com) Manager 15-MAR-17 102  
 203 Chris Davis [cdavis@example.com](mailto:cdavis@example.com) Clerk 22-JUN-19 103  
 204 Diana Morris [dmorris@example.com](mailto:dmorris@example.com) Security 10-JAN-20 104  
 205 Ethan Scott [escott@example.com](mailto:escott@example.com) Teller 12-AUG-16 105  
 206 Fiona Hill [fhill@example.com](mailto:fhill@example.com) Clerk 27-FEB-15 106  
 207 George Young [gyoung@example.com](mailto:gyoung@example.com) Manager 03-NOV-14 107  
 208 Hannah Green [hgreen@example.com](mailto:hgreen@example.com) Clerk 18-MAY-21 108  
 209 Ian Lopez [ilopez@example.com](mailto:ilopez@example.com) Teller 09-SEP-13 109  
 210 Judy Turner [jturner@example.com](mailto:jturner@example.com) Manager 12-DEC-12 110

10 rows selected.

SQL> SQL> --Query 6: Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 10 rows SQL> -- Business purpose: this query joins information of attributes from customer, account and transaction limited to five attributes each and the first ten rows. SQL> select c.f\_name as first\_name, c.l\_name as last\_name, c.address, c.ph\_number as phone\_number, c.email, 2 a.account\_id as account\_number, a.balance, a.acct\_type as account\_type, a.creat\_date as creation\_date, a.status, 3 t.transc\_id as transaction\_number, t.amount, t.transc\_type as transaction\_type, t.timestamp, t.description 4 from 5 customer c join account a on c.cust\_id = a.fk\_customer\_cust\_id join transaction t on a.account\_id = t.fk\_account\_account\_id 6 fetch first 10 rows only;

FIRST\_NAME LAST\_NAME ADDRESS PHONE\_NUMB EMAIL ACCOUNT\_NUMBER BALANCE ACCOUNT\_TYPE CREATION\_ STATUS TRANSACTION\_NUMBER AMOUNT TRANSACTION\_TYPE TIMESTAMP DESCRIPTION

Alicia Mills 19 Garden Ave, Austin 5125551001 [alicia.mills@example.com](mailto:alicia.mills@example.com) 401 2300.5 Savings 10-JAN-22 Active 601 200 Deposit 05-AUG-25 06.05.08.000000000 AM Initial deposit

Alicia Mills 19 Garden Ave, Austin 5125551001 [alicia.mills@example.com](mailto:alicia.mills@example.com) 425 510 Savings 08-AUG-23 Active 615 600 Transfer 05-AUG-25 06.05.08.000000000 AM Loan repayment

Brian Tucker 82 Ridge Rd, Denver 3035554521 [brian.tucker@example.com](mailto:brian.tucker@example.com) 402 150.75 Checking 22-MAR-22 Active 602 100 Withdrawal 05-AUG-25 06.05.08.000000000 AM ATM withdrawal

Clara Nguyen 34 Willow St, Seattle 2065553122 [clara.nguyen@example.com](mailto:clara.nguyen@example.com) 403 6700 Loan 05-SEP-21 Closed 603 300 Transfer 05-AUG-25 06.05.08.000000000 AM Transfer to savings

Clara Nguyen 34 Willow St, Seattle 2065553122 [clara.nguyen@example.com](mailto:clara.nguyen@example.com) 403 6700 Loan 05-SEP-21 Closed 606 25 Withdrawal 05-AUG-25 06.05.08.000000000 AM ATM fee

Clara Nguyen 34 Willow St, Seattle 2065553122 [clara.nguyen@example.com](mailto:clara.nguyen@example.com) 403 6700 Loan 05-SEP-21 Closed 609 120 Deposit 05-AUG-25 06.05.08.000000000 AM Cash deposit

Clara Nguyen 34 Willow St, Seattle 2065553122 [clara.nguyen@example.com](mailto:clara.nguyen@example.com) 423 543.21 Checking 14-FEB-23 Frozen 612 150 Withdrawal 05-AUG-25 06.05.08.000000000 AM ATM withdrawal

Clara Nguyen 34 Willow St, Seattle 2065553122 [clara.nguyen@example.com](mailto:clara.nguyen@example.com) 423 543.21 Checking 14-FEB-23 Frozen 614 75 Withdrawal 05-AUG-25 06.05.08.000000000 AM POS purchase

Derek Ford 106 Main Blvd, Miami 7865559982 [derek.ford@example.com](mailto:derek.ford@example.com) 404 980.2 Savings 13-JUL-22 Active 604 50 Withdrawal 05-AUG-25 06.05.08.000000000 AM POS Purchase

Ella Brown 501 Pacific St, Portland 5035557411 [ella.brown@example.com](mailto:ella.brown@example.com) 405 3120 Checking 01-JAN-23 Active 605 1000 Deposit 05-AUG-25 06.05.08.000000000 AM Salary

10 rows selected.

SQL> SQL> --Query 7: Select distinct rows using joins on 3 tables SQL> -- Business purpose: select all distinct information from branches, employees, operation after joining them. SQL> select distinct \* 2 from branch b join employee e on b.br\_id = e.fk\_branch\_br\_id join operation o on b.br\_id = o.fk\_branch\_br\_id;

BR\_ID BR\_NAME LOCATION MGR\_NAME PH\_NUMBER EST\_DATE EMP\_ID F\_NAME L\_NAME EMAIL POSITION HIRED\_DAT FK\_BRANCH\_BR\_ID DATE\_CREATED CREATED\_BY CHANNEL APPR\_CODE OP\_STATUS FK\_BRANCH\_BR\_ID FK\_TRANSACTION\_TRANSC\_ID

101 Downtown New York Alice Miles 2125551010 01-JAN-10 201 Angela White [awhite@example.com](mailto:awhite@example.com) Teller 01-APR-18 101 05-AUG-25 06.05.08.000000000 AM Angela White Teller AP123 Approved 101 601  
 101 Downtown New York Alice Miles 2125551010 01-JAN-10 201 Angela White [awhite@example.com](mailto:awhite@example.com) Teller 01-APR-18 101 05-AUG-25 06.05.08.000000000 AM Angela White ATM AP133 Approved 101 611  
 102 Midtown Chicago Brian Lee 3125551020 15-MAY-12 202 Ben King [bking@example.com](mailto:bking@example.com) Manager 15-MAR-17 102 05-AUG-25 06.05.08.000000000 AM Ben King ATM AP124 Completed 102 602  
 102 Midtown Chicago Brian Lee 3125551020 15-MAY-12 202 Ben King [bking@example.com](mailto:bking@example.com) Manager 15-MAR-17 102 05-AUG-25 06.05.08.000000000 AM Ben King Online AP134 Completed 102 612  
 103 Uptown Houston Carla Yu 7135551030 20-MAR-14 203 Chris Davis [cdavis@example.com](mailto:cdavis@example.com) Clerk 22-JUN-19 103 05-AUG-25 06.05.08.000000000 AM Chris Davis Online AP125 Approved 103 603  
 103 Uptown Houston Carla Yu 7135551030 20-MAR-14 203 Chris Davis [cdavis@example.com](mailto:cdavis@example.com) Clerk 22-JUN-19 103 05-AUG-25 06.05.08.000000000 AM Chris Davis Teller AP135 Approved 103 613  
 104 Central Los Angeles David Singh 2135551040 10-AUG-11 204 Diana Morris [dmorris@example.com](mailto:dmorris@example.com) Security 10-JAN-20 104 05-AUG-25 06.05.08.000000000 AM Diana Morris Teller AP126 Rejected 104 604  
 104 Central Los Angeles David Singh 2135551040 10-AUG-11 204 Diana Morris [dmorris@example.com](mailto:dmorris@example.com) Security 10-JAN-20 104 05-AUG-25 06.05.08.000000000 AM Diana Morris ATM AP136 Completed 104 614  
 105 Westside San Diego Eva Knight 6195551050 01-SEP-13 205 Ethan Scott [escott@example.com](mailto:escott@example.com) Teller 12-AUG-16 105 05-AUG-25 06.05.08.000000000 AM Ethan Scott ATM AP127 Approved 105 605  
 105 Westside San Diego Eva Knight 6195551050 01-SEP-13 205 Ethan Scott [escott@example.com](mailto:escott@example.com) Teller 12-AUG-16 105 05-AUG-25 06.05.08.000000000 AM Ethan Scott Online AP137 Approved 105 615  
 106 Southpoint Miami Fred Owen 3055551060 12-OCT-15 206 Fiona Hill [fhill@example.com](mailto:fhill@example.com) Clerk 27-FEB-15 106 05-AUG-25 06.05.08.000000000 AM Fiona Hill Online AP128 Approved 106 606  
  
 BR\_ID BR\_NAME LOCATION MGR\_NAME PH\_NUMBER EST\_DATE EMP\_ID F\_NAME L\_NAME EMAIL POSITION HIRED\_DAT FK\_BRANCH\_BR\_ID DATE\_CREATED CREATED\_BY CHANNEL APPR\_CODE OP\_STATUS FK\_BRANCH\_BR\_ID FK\_TRANSACTION\_TRANSC\_ID

106 Southpoint Miami Fred Owen 3055551060 12-OCT-15 206 Fiona Hill [fhill@example.com](mailto:fhill@example.com) Clerk 27-FEB-15 106 05-AUG-25 06.05.08.000000000 AM Fiona Hill Teller AP138 Rejected 106 601  
 107 Riverside Denver Grace Patel 3035551070 21-JUL-16 207 George Young [gyoung@example.com](mailto:gyoung@example.com) Manager 03-NOV-14 107 05-AUG-25 06.05.08.000000000 AM George Young Teller AP129 Completed 107 607  
 107 Riverside Denver Grace Patel 3035551070 21-JUL-16 207 George Young [gyoung@example.com](mailto:gyoung@example.com) Manager 03-NOV-14 107 05-AUG-25 06.05.08.000000000 AM George Young ATM AP139 Pending 107 602  
 108 Northend Seattle Henry Kim 2065551080 14-FEB-18 208 Hannah Green [hgreen@example.com](mailto:hgreen@example.com) Clerk 18-MAY-21 108 05-AUG-25 06.05.08.000000000 AM Hannah Green ATM AP130 Approved 108 608  
 108 Northend Seattle Henry Kim 2065551080 14-FEB-18 208 Hannah Green [hgreen@example.com](mailto:hgreen@example.com) Clerk 18-MAY-21 108 05-AUG-25 06.05.08.000000000 AM Hannah Green Online AP140 Completed 108 603  
 109 Eastside Atlanta Isla Brown 4045551090 25-APR-17 209 Ian Lopez [ilopez@example.com](mailto:ilopez@example.com) Teller 09-SEP-13 109 05-AUG-25 06.05.08.000000000 AM Ian Lopez Online AP131 Approved 109 609  
 109 Eastside Atlanta Isla Brown 4045551090 25-APR-17 209 Ian Lopez [ilopez@example.com](mailto:ilopez@example.com) Teller 09-SEP-13 109 05-AUG-25 06.05.08.000000000 AM Ian Lopez Teller AP141 Approved 109 604  
 110 Metro Boston Jack Chen 6175551100 31-DEC-19 210 Judy Turner [jturner@example.com](mailto:jturner@example.com) Manager 12-DEC-12 110 05-AUG-25 06.05.08.000000000 AM Judy Turner Teller AP132 Pending 110 610  
 110 Metro Boston Jack Chen 6175551100 31-DEC-19 210 Judy Turner [jturner@example.com](mailto:jturner@example.com) Manager 12-DEC-12 110 05-AUG-25 06.05.08.000000000 AM Judy Turner ATM AP142 Approved 110 605

20 rows selected.

SQL> SQL> --Query 8: Use GROUP BY and HAVING in a select statement using one or more tables SQL> -- Business purpose: select all information from accounts that are grouped by account type where the totals by per account type is greater than 1000.00 SQL> select acct\_type as account\_type, sum(balance) as total\_balance from account group by acct\_type having sum(balance) > 1000.00;

ACCOUNT\_TYPE TOTAL\_BALANCE

Savings 8496.64 Checking 6216.76 Loan 58900

SQL> SQL> --Query 9: Use IN clause to select data from one or more tables SQL> -- Business purpose: select all information of three attributes after joining accounts and transactions. SQL> select a.balance, a.acct\_type as account\_type, a.status, t.amount, t.transc\_type as transaction\_type, t.description 2 from account a join transaction t on a.account\_id = t.fk\_account\_account\_id 3 where transc\_type in ('Deposit', 'Withdrawal');

BALANCE ACCOUNT\_TYPE STATUS AMOUNT TRANSACTION\_TYPE DESCRIPTION

2300.5 Savings Active 200 Deposit Initial deposit   
150.75 Checking Active 100 Withdrawal ATM withdrawal   
 6700 Loan Closed 25 Withdrawal ATM fee   
 6700 Loan Closed 120 Deposit Cash deposit   
 980.2 Savings Active 50 Withdrawal POS Purchase   
 3120 Checking Active 1000 Deposit Salary

1125.35 Savings Active 500 Deposit Freelance payment

220.1 Savings Active 200 Withdrawal Bill payment

4200 Loan Closed 500 Deposit Initial deposit

543.21 Checking Frozen 150 Withdrawal ATM withdrawal

543.21 Checking Frozen 75 Withdrawal POS purchase

11 rows selected.

SQL> SQL> --Query 10: Select length of one column from one table (use LENGTH function) SQL> -- Business purpose: this query calculates the length of the name of employees who execute transactions. SQL> select length(created\_by) as name\_length from operation;

## **NAME\_LENGTH**

12  
 12  
 8  
 8  
 11  
 11  
 12  
 12  
 11  
 11  
 10

## **NAME\_LENGTH**

10  
 12  
 12  
 12  
 12  
 9  
 9  
 11  
 11

20 rows selected.

SQL> SQL> --Query 11: Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement. SQL> -- Make sure you use ROLLBACK afterwards so that the data will not be physically removed SQL> -- Business purpose: this query delete a record of information from customer data, then rollback the deleted record. SQL> select \* from customer;

CUST\_ID F\_NAME L\_NAME DOB ADDRESS PH\_NUMBER EMAIL

1 Alicia Mills 24-MAY-91 19 Garden Ave, Austin 5125551001 [alicia.mills@example.com](mailto:alicia.mills@example.com)   
 2 Brian Tucker 07-NOV-88 82 Ridge Rd, Denver 3035554521 [brian.tucker@example.com](mailto:brian.tucker@example.com)   
 3 Clara Nguyen 16-FEB-95 34 Willow St, Seattle 2065553122 [clara.nguyen@example.com](mailto:clara.nguyen@example.com)   
 4 Derek Ford 12-SEP-90 106 Main Blvd, Miami 7865559982 [derek.ford@example.com](mailto:derek.ford@example.com)   
 5 Ella Brown 04-MAR-86 501 Pacific St, Portland 5035557411 [ella.brown@example.com](mailto:ella.brown@example.com)   
 6 Frank Lopez 19-JUL-93 212 Walnut Ln, Chicago 7735552188 [frank.lopez@example.com](mailto:frank.lopez@example.com)   
 7 Grace Kim 02-DEC-89 45 Sunset Ave, LA 2135553011 [grace.kim@example.com](mailto:grace.kim@example.com)   
 8 Henry Singh 28-JUN-92 87 Liberty St, Boston 6175559003 [henry.singh@example.com](mailto:henry.singh@example.com)   
 9 Isla Patel 10-AUG-94 15 Beacon Way, New York 7185556020 [isla.patel@example.com](mailto:isla.patel@example.com)   
 10 Jack Davis 05-OCT-96 68 Pine Circle, Dallas 2145557777 [jack.davis@example.com](mailto:jack.davis@example.com)

10 rows selected.

SQL> delete from customer where cust\_id = 1;

1 row deleted.

SQL> select \* from customer;

CUST\_ID F\_NAME L\_NAME DOB ADDRESS PH\_NUMBER EMAIL

2 Brian Tucker 07-NOV-88 82 Ridge Rd, Denver 3035554521 [brian.tucker@example.com](mailto:brian.tucker@example.com)   
 3 Clara Nguyen 16-FEB-95 34 Willow St, Seattle 2065553122 [clara.nguyen@example.com](mailto:clara.nguyen@example.com)   
 4 Derek Ford 12-SEP-90 106 Main Blvd, Miami 7865559982 [derek.ford@example.com](mailto:derek.ford@example.com)   
 5 Ella Brown 04-MAR-86 501 Pacific St, Portland 5035557411 [ella.brown@example.com](mailto:ella.brown@example.com)   
 6 Frank Lopez 19-JUL-93 212 Walnut Ln, Chicago 7735552188 [frank.lopez@example.com](mailto:frank.lopez@example.com)   
 7 Grace Kim 02-DEC-89 45 Sunset Ave, LA 2135553011 [grace.kim@example.com](mailto:grace.kim@example.com)   
 8 Henry Singh 28-JUN-92 87 Liberty St, Boston 6175559003 [henry.singh@example.com](mailto:henry.singh@example.com)   
 9 Isla Patel 10-AUG-94 15 Beacon Way, New York 7185556020 [isla.patel@example.com](mailto:isla.patel@example.com)   
 10 Jack Davis 05-OCT-96 68 Pine Circle, Dallas 2145557777 [jack.davis@example.com](mailto:jack.davis@example.com)

9 rows selected.

SQL> rollback;

Rollback complete.

SQL> SQL> --Query 12: Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement. SQL> -- Make sure you use ROLLBACK afterwards so that the data will not be physically removed SQL> -- Business purpose: this query changes the first name and last name of the record that has cutomer id set to 1 from customer data, then rollback to initial values. SQL> select \* from customer;

CUST\_ID F\_NAME L\_NAME DOB ADDRESS PH\_NUMBER EMAIL

1 Alicia Mills 24-MAY-91 19 Garden Ave, Austin 5125551001 [alicia.mills@example.com](mailto:alicia.mills@example.com)   
 2 Brian Tucker 07-NOV-88 82 Ridge Rd, Denver 3035554521 [brian.tucker@example.com](mailto:brian.tucker@example.com)   
 3 Clara Nguyen 16-FEB-95 34 Willow St, Seattle 2065553122 [clara.nguyen@example.com](mailto:clara.nguyen@example.com)   
 4 Derek Ford 12-SEP-90 106 Main Blvd, Miami 7865559982 [derek.ford@example.com](mailto:derek.ford@example.com)   
 5 Ella Brown 04-MAR-86 501 Pacific St, Portland 5035557411 [ella.brown@example.com](mailto:ella.brown@example.com)   
 6 Frank Lopez 19-JUL-93 212 Walnut Ln, Chicago 7735552188 [frank.lopez@example.com](mailto:frank.lopez@example.com)   
 7 Grace Kim 02-DEC-89 45 Sunset Ave, LA 2135553011 [grace.kim@example.com](mailto:grace.kim@example.com)   
 8 Henry Singh 28-JUN-92 87 Liberty St, Boston 6175559003 [henry.singh@example.com](mailto:henry.singh@example.com)   
 9 Isla Patel 10-AUG-94 15 Beacon Way, New York 7185556020 [isla.patel@example.com](mailto:isla.patel@example.com)   
 10 Jack Davis 05-OCT-96 68 Pine Circle, Dallas 2145557777 [jack.davis@example.com](mailto:jack.davis@example.com)

10 rows selected.

SQL> update customer set f\_name = 'Donald', l\_name = 'Keita' where cust\_id = 2;

1 row updated.

SQL> select \* from customer;

CUST\_ID F\_NAME L\_NAME DOB ADDRESS PH\_NUMBER EMAIL

1 Alicia Mills 24-MAY-91 19 Garden Ave, Austin 5125551001 [alicia.mills@example.com](mailto:alicia.mills@example.com)   
 2 Donald Keita 07-NOV-88 82 Ridge Rd, Denver 3035554521 [brian.tucker@example.com](mailto:brian.tucker@example.com)   
 3 Clara Nguyen 16-FEB-95 34 Willow St, Seattle 2065553122 [clara.nguyen@example.com](mailto:clara.nguyen@example.com)   
 4 Derek Ford 12-SEP-90 106 Main Blvd, Miami 7865559982 [derek.ford@example.com](mailto:derek.ford@example.com)   
 5 Ella Brown 04-MAR-86 501 Pacific St, Portland 5035557411 [ella.brown@example.com](mailto:ella.brown@example.com)   
 6 Frank Lopez 19-JUL-93 212 Walnut Ln, Chicago 7735552188 [frank.lopez@example.com](mailto:frank.lopez@example.com)   
 7 Grace Kim 02-DEC-89 45 Sunset Ave, LA 2135553011 [grace.kim@example.com](mailto:grace.kim@example.com)   
 8 Henry Singh 28-JUN-92 87 Liberty St, Boston 6175559003 [henry.singh@example.com](mailto:henry.singh@example.com)   
 9 Isla Patel 10-AUG-94 15 Beacon Way, New York 7185556020 [isla.patel@example.com](mailto:isla.patel@example.com)   
 10 Jack Davis 05-OCT-96 68 Pine Circle, Dallas 2145557777 [jack.davis@example.com](mailto:jack.davis@example.com)

10 rows selected.

SQL> rollback;

Rollback complete.

SQL> SQL> --Perform 8 Additional Advanced Queries SQL> SQL> -- Query 13: use GROUP BY count the number of branch in each location from view called "view\_br\_performance" SQL> -- Business purpose: this query computes the number of branch by location from already created view, "view\_br\_performance" SQL> select location, count(br\_name) as "BRANCHES PER LOCATION" from view\_br\_performance group by location ;

LOCATION BRANCHES PER LOCATION

Miami 1 Los Angeles 2 San Diego 1 Atlanta 1 Denver 2 Chicago 1 Boston 2 New York 2 Seattle 1 Houston 1

10 rows selected.

SQL> SQL> -- Query 14: use GROUP BY to show the number transaction per customer SQL> -- Business purpose: this query compute the number of transaction carried out by customer SQL> select c.f\_name as "First Name", c.l\_name as "Last Name", count(transc\_id) as "Number of Transaction" 2 from customer c join account a on c.cust\_id = a.fk\_customer\_cust\_id join transaction t on a.account\_id = t.fk\_account\_account\_id 3 group by c.f\_name, c.l\_name;

First Name Last Name Number of Transaction

Alicia Mills 2 Brian Tucker 1 Clara Nguyen 5 Derek Ford 1 Ella Brown 2 Grace Kim 1 Henry Singh 1 Isla Patel 1 Jack Davis 1

9 rows selected.

SQL> SQL> -- Query 15:show the five active account that were lastly created ORDER BY and FIRST n ROWS ONLY SQL> -- Business purpose: this query select the last five created account that are still active SQL> select account\_id as "Account number", creat\_date as "Creation Date", status as "Status" 2 from account where status = 'Active' order by creat\_date desc fetch first 5 rows only;

Account number Creation Status

425 08-AUG-23 Active   
 419 01-JUL-23 Active   
 413 20-JUN-23 Active   
 421 23-MAY-23 Active   
 416 12-APR-23 Active

SQL> SQL> -- Query 16: Show the five first customers who two account by using JOIN and operator NOT IN. SQL> -- Business purpose: this query show information about customer who have both Checking and Savings accounts SQL> select c.f\_name || ' ' || c.l\_name as "Customer Name", account\_id as "Account number", a.acct\_type as "Account Type" 2 from customer c join account a on c.cust\_id = a.fk\_customer\_cust\_id 3 where a.acct\_type not in ('Loan');

Customer Name Account number Account Type

Alicia Mills 401 Savings

Alicia Mills 411 Checking

Alicia Mills 425 Savings

Brian Tucker 402 Checking

Brian Tucker 416 Savings

Brian Tucker 422 Savings

Clara Nguyen 413 Savings

Clara Nguyen 423 Checking

Derek Ford 404 Savings

Derek Ford 414 Checking

Derek Ford 417 Checking

Customer Name Account number Account Type

Ella Brown 405 Checking

Grace Kim 407 Savings

Henry Singh 408 Checking

Henry Singh 419 Savings

Isla Patel 420 Checking

Jack Davis 410 Savings

17 rows selected.

SQL> SQL> -- Query 17: use JOIN and HAVING to aggregate the total amount of transaction per account. SQL> -- Business purpose: this query the total amount of transaction per account with a total of more $500. SQL> select a.account\_id, sum(t.amount) 2 from account a join transaction t on a.account\_id = t.fk\_account\_account\_id 3 group by a.account\_id having sum(t.amount) > 500;

ACCOUNT\_ID SUM(T.AMOUNT)

405 1000  
 425 600

SQL> SQL> -- Query 18: use LEFT JOIN to show all information all transaction and those done at branch SQL> -- Business purpose: this script show transaction information along with the one that SQL> select tp.fk\_account\_account\_id as "ACCOUNT NUMBER", tp.amount, tp.description, tp.channel 2 from 3 (select t.fk\_account\_account\_id, t.amount, t.description, o.channel 4 from transaction t left join operation o on t.transc\_id = o.fk\_transaction\_transc\_id 5 where o.channel in ('ATM', 'Teller')) tp;

ACCOUNT NUMBER AMOUNT DESCRIPTION CHANNEL

401 200 Initial deposit Teller   
 401 200 Initial deposit Teller   
 402 100 ATM withdrawal ATM   
 402 100 ATM withdrawal ATM   
 404 50 POS Purchase Teller   
 404 50 POS Purchase Teller   
 405 1000 Salary ATM   
 405 1000 Salary ATM   
 407 500 Freelance payment Teller   
 408 75 Transfer to friend ATM   
 410 200 Bill payment Teller

ACCOUNT NUMBER AMOUNT DESCRIPTION CHANNEL

415 500 Initial deposit ATM   
 420 300 Transfer to savings Teller   
 423 75 POS purchase ATM

14 rows selected.

SQL> SQL> -- Query 19: use SUBQUERY to show the name and location who carried out operations that have "Rejected" status SQL> -- Business purpose:this query select employees that assisted customers resulting in rejected operations and the locations and branches where they work SQL> select e.f\_name as "First Name", e.l\_name as "Last Name", 2 (select b.location 3 from branch b 4 where b.br\_id = e.fk\_branch\_br\_id) as "Branch Location" 5 from employee e 6 where e.f\_name || ' ' || e.l\_name in ( 7 select o.created\_by 8 from operation o 9 where o.op\_status = 'Rejected' 10 );

First Name Last Name Branch Location

Diana Morris Los Angeles

Fiona Hill Miami

SQL> SQL> -- Query 20: use JOIN Customer, Account, Transaction and Operation tables to show customers who carried out operation where SQL> -- transaction\_type is "Withdrawal" and channel is "ATM" SQL> -- Business purpose: this query select all customers that withdraw money from ATM machines SQL> select c.f\_name || ' ' || c.l\_name as "Customer Name", t.transc\_type as "Transaction Type", o.channel as "Channel", t.amount as "Amount" 2 from customer c join account a on c.cust\_id = a.fk\_customer\_cust\_id join transaction t on a.account\_id = t.fk\_account\_account\_id 3 join operation o on t.transc\_id = o.fk\_transaction\_transc\_id 4 where t.transc\_type = 'Withdrawal' and o.channel = 'ATM';

Customer Name Transaction Type Channel Amount

Brian Tucker Withdrawal ATM 100

Brian Tucker Withdrawal ATM 100

Clara Nguyen Withdrawal ATM 75

**9. Database Administration and Monitoring**

This section describes how the BTMS database will be maintained and monitored to ensure optimal performance, data security, and availability. Key administrative practices include role definitions, backup strategies, and monitoring protocols.

9.1 Roles and Responsibilities

* Database Administrator (DBA): Responsible for installation, configuration, performance tuning, and maintenance of the BTMS database.
* System Architect: Ensures design alignment with performance and security expectations.
* Developers: Implement DDL, DML, views, and sequences.
* End Users: Responsible for executing application queries and using system outputs for decision-making.

9.2 Availability and Recovery

Backups will be scheduled weekly with incremental backups daily. The system will utilize Oracle's recovery manager for data restoration. Audit columns and triggers help track all changes and support rollback strategies when necessary.

9.3 Monitoring and Performance

Oracle SQL Developer tools will be used for real-time performance monitoring, including query optimization and indexing analysis. Views such as 'view\_cust\_acct\_summary' and 'view\_br\_performance' are critical for identifying performance bottlenecks at the customer and branch levels.

9.4 Security and Privacy Considerations

Sensitive operations are abstracted using views. The database enforces strong foreign key relationships and uses surrogate keys. Audit columns track user actions, helping ensure data integrity and traceability. Access control will be implemented via role-based privileges. However, DCL is out of scope for this project.